Feature Selection and Classification Using Age Layered Population Structure Genetic Programming

by
Anthony Awuley

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Master of Science

Department of Computer Science
Brock University, St. Catharines, Ontario

Abstract

The curse of dimensionality is a major problem in the fields of machine learning, data mining and knowledge discovery. Exhaustive search for the most optimal subset of relevant features from a high dimensional dataset is NP hard. Sub–optimal population based stochastic algorithms such as GP and GA are good choices for searching through large search spaces, and are usually more feasible than exhaustive and deterministic search algorithms. On the other hand, population based stochastic algorithms often suffer from pre–mature convergence on mediocre sub–optimal solutions. The Age Layered Population Structure (ALPS) is a novel meta–heuristic for overcoming the problem of premature convergence in evolutionary algorithms, and for improving search in the fitness landscape. The ALPS paradigm uses an age–measure to control breeding and competition between individuals in the population. This thesis uses a modification of the ALPS GP strategy called Feature Selection ALPS (FSALPS) for feature subset selection and classification of varied supervised learning tasks. FSALPS uses a novel frequency count system to rank features in the GP population based on evolved feature frequencies. The ranked features are translated into probabilities, which are used to control evolutionary processes such as terminal–symbol selection for the construction of GP trees/sub-trees. The FSALPS meta–heuristic continuously refines the feature subset selection process whiles simultaneously evolving efficient classifiers through a non–converging evolutionary process that favors selection of features with high discrimination of class labels. We investigated and compared the performance of canonical GP, ALPS and FSALPS on high–dimensional benchmark classification datasets, including a hyperspectral image. Using Tukey's HSD ANOVA test at a 95% confidence interval, ALPS and FSALPS dominated canonical GP in evolving smaller but efficient trees with less bloat expressions. FSALPS significantly outperformed canonical GP and ALPS and some reported feature selection strategies in related literature on dimensionality reduction.

Acknowledgements

I would like to thank the following individuals and organizations for their tireless support:

1. Brian J. Ross for accepting me as a student and providing two years of excellent supervision, guidance, and funding.I could not have gotten a better supervisor.

2. Beatrice Ombuki–Berman and Sheridan Houghten for their participation on the supervisory committee.

3. Cale Fairchild for providing an outstanding technical support and knowledge on computing clusters and Linux expertise.

4. The Computer Science department and Brock Graduate School for providing me with the privilege, facilities, funding and enabling environment to accomplish something great.

Love is patient and kind. Love is not jealous. It does not brag, does not get puffed up, does not behave indecently, does not look for its own interests, does not become provoked. It does not keep account of the injury. It does not rejoice over unrighteousness, but rejoices with the truth. It bears all things, belives all things, hopes all things, endures all things. Love never fails.

1 Corinthians 13:4-8

A.A

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning (ML) is a branch of artificial intelligence that deals with automatic induction of knowledge from data [54]. The 21^{st} century has seen an exponential growth[50] of high–dimensional data in bioinformatics, web/text data, biology, medicine, finance etc., necessitating the use of ML techniques such as GP and GA for data analysis. The pervasive nature of high dimensional data challenges present techniques used in data mining, pattern recognition and information filtering. Model construction in ML is used to induce hypotheses that can be learned from data. A hypothesis is a function that predicts classes from input features[50] and is referred to here as a classifier. High dimensional problems have large hypothesis space that usually results in an increased difficulty in selecting the most efficient hypothesis. Classification is one of the major tasks in data mining, and has to do with the use of features or attributes to predict class labels (outputs or targets)[63]. In [9], a feature is defined as "an independent measurable property of a process been observed". In ML, features are used to generate models for classification. The number of features considered for a given problem has a direct bearing on the hypothesis space [54]. A linear growth in the number of features results in an exponential grow in the hypothesis space therefore a reduced feature vector reduces the hypothesis space and increases the chance of discovering the most appropriate hypothesis. Feature selection (FS) is used to reduce the dimensionality of a given dataset. According to [61], relevant feature subset selection for a learning systems improve understanding of the data, leads to the design of better classifier models, eliminate irrelevant data with benefits such as enhanced data visualization, reduced computational cost for constructing learning models and improve generalization of constructed models. Feature extraction (FE) on the other hand, deals with the discovery of composite features from an original feature vector that are more representative of the dataset. The newly discovered fea-

tures (linear or non-linear combination of the original features) should score a higher prediction rate of the class labels. FS and FE are key preprocessing steps in machine learning.

Given that GP has a dynamic expressive ability, it can be guided using a fitness function to represent solutions for different problem domains. A GP classifier is an evolved GP expression that measures how a selected subset of features from the original feature vectors predicts the class label(s) for some data instances.

Evolutionary algorithms often converge on suboptimal solutions, as is the case for most stochastic algorithms. The problem of finding and optimal set of feature vectors out of an original set is NP complete [50, 9]. Feature reduction to a set of relevant features directly benefits machine–learning algorithms.

The age layered population structure (ALPS) strategy is used to overcome the problem of premature convergence in algorithms with elements of randomness in them. In ALPS, the regular introduction of new individuals into the population results in an evolutionary algorithm that is almost never converged but constantly exploring different parts of the fitness landscape [29] with an increased chance of landing on a global optimum solution. A number of published works have recorded the benefit of the ALPS system [29, 30, 59].

This research presents feature selection and ALPS (FSALPS) as a novel GP system that relies on the ALPS strategy to perform directed feature selection. FSALPS seeks to advance GPs ability to perform automatic feature selection by integrating knowledge of feature selection in the ALPS layers to control generation of new individuals and genetic operations. In FSALPS, features with high discrimination of class labels are favored without forcing the EA system in premature convergence or limiting the ability of the EA system to explore relevant parts of the fitness landscape.

1.1 Goals and Motivation

The success of ML in database and knowledge discovery (DKD) tasks is largely dependent on the number of features or attributes considered for a problem[9]. An inefficient selection of input data will limit performance of machine learning tasks while a carefully selected data will significantly improve performance and enhance accuracy of prediction models. This explains why the number of variables considered for a learning system affects the extent to which meaningful knowledge can be discovered. Most real world problems have huge volumes of data with noisy variables (objectives). High dimensional data increases the hypothesis space, reduces the

classification accuracy of model construction algorithms and introduces the curse of dimensionality [50]. This can be overcome by applying preprocessing techniques in machine learning such as feature selection or feature extraction. The selective refinement of feature variables involves elimination of noisy variables leaving a set of relevant variables that have significant impact on the measured process. An n feature vector problem has an exponential solution space of 2^n. A reduction of the original feature vector by k reduces the solution space to $2^n/2^k$. A reduced feature set to relevant feature values leads to improved prediction accuracy and better tractability for large dimensional datasets. Reducing curse of dimensionality for DKD problems to computationally feasible problems open a new door to the discovery of meaningful knowledge from such datasets. This will advance ML in most research areas and could easily lead to potential breakthroughs in many application fields.

Population based heuristic search algorithms such as GA and GP have been used in classification problems in feature selection and construction. Due to its representational superiority, GP is able to perform automatic feature selection while evolving a classifier. The evolutionary process favors GP tree expression that select a subset of feature variables (terminal symbols) that maximize the prediction accuracy of the classifier. The feature selection ability of GP is directly challenged by possible occurrence of bloat expressions in evolved individuals. Features in bloat expressions increase the overall score of frequency count of such features and might be misleading towards measuring feature relevance. Bloat expressions in canonical GP multiply as individual solutions increase in size due to genetic operations with a corresponding increase in space requirements as well as initialization and evaluation time.

We propose a novel evolutionary feature selection and classification strategy that uses the ALPS algorithm to evolve a classifier and perform feature subset selection with high classification accuracy. The approach is achieved by using ALPS to overcome the problem of premature convergence in canonical GP and simultaneously evolve feature subsets with high discrimination of class labels. Our proposed FSALPS strategy performs feature selection using ALPS GP towards dimensionality reduction. FSALPS uses a novel frequency–based feature–ranking system that will be used to control domain specific genetic operations and to determine probability of feature assignment to new randomly created individuals. The proposed meta–heuristic will be tested on high dimensional benchmark datasets and compared to canonical GP, ALPS GP and other statistical strategies used in classification, feature selection and construction. Combining ALPS search power with GPs dynamic tree encoding to perform feature subset selection will lead to a significant improvement in aspects

of classification problems involving high dimensional datasets. We will show that FSALPS GP significantly reduces the number of features whiles maintaining or improving classification accuracy. We will simultaneously exploit GPs default encoding for the construction of a classifier system to measure performance of the selected feature subset. This will effectively reduce the computational effort needed to design a classifier system as compared to other feature selection strategies that rely on external classifiers.

1.2 Thesis Structure

Chapter 2 reviews relevant background information on topics such as classification, feature selection, heuristic and population based algorithms, genetic programming and the ALPS strategy. We will continue with a literature review of related research in feature selection in Chapter 3. Chapter 4 builds on the ALPS algorithm by introducing a feature selection paradigm named FSALPS. In Chapter 5, we will discuss and compare the performance of canonical GP, ALPS GP and FSALPS strategy on feature selection and classifier construction using multiple ML datasets. We will also examine a high dimensional hyperspectral data to provide further evidence of the feature selection ability of FSALPS in Chapter 6. In Chapter 7, FSALPS is compared to related works. Concluding remarks and future research is presented in Chapter 8.

Chapter 2

Background

2.1 Introduction

In the fields of pattern recognition, data mining and knowledge discovery, feature reduction influences the quality of learning models constructed from high dimensional data. Feature selection eliminates features that offer little or no contribution to the class label. For instance, a demographic data that is to be used to predict the occurrence of a particular disease within a population may contain features or attributes that are irrelevant to the disease condition. The removal of such irrelevant or noisy attributes will enhance performance of learning models and significantly reduce the hypothesis space needed for model construction. Feature selection has been used as a pre–processing technique for function approximation, classification and clustering, using learning systems designed in neural networks[8], regression models[10] and decision trees[16]. A number of classification algorithms have been used in feature selection and feature construction.

In this thesis, we will compare existing and new evolutionary techniques for feature selection and classification on supervised learning tasks. A number of methods have been developed for feature selection or variable elimination and are broadly classified into filter, wrapper and embedded methods[50]. Feature reduction as studied in machine learning can be broadly classified into two main problem domains –supervised and unsupervised learning.

2.2 Classification

Classification is a major task in supervised machine learning [54] and has been applied to a wide range of problems such as credit scoring, bankruptcy prediction, medical diagnosis, pattern recognition, text categorization, and software quality assessment. [20]. The goal of classification is to accurately predict discrete class labels using predictor values or feature variables. Prediction rules are applied to pre–classified training examples such that the rule with the highest prediction accuracy with the ability to generalize other unseen test examples is preferred. Classification data could be as simple as in binary classification, where the target values to be predicted are two e.g. high credit rating or low credit rating, or diabetic or non–diabetic. It could also be as complex as multi class classification involving identification of numerous class labels as seen in multiple target identification problems. In [57] an example of multi classification is given as case where a loan applicant could fall within any of the following credit ratings: low, medium, or high credit risks. The credit rating for each client will be treated as the target while the other attributes (e.g. home ownership or rental and/or location of apartment/building, number of years of rent, type of employment, investments, borrowing records etc.) will be treated as the predictors [or features]. Given the above problem, a classification algorithm uses a pre–classified training example to generate a highly generalizable rule to relate most/all features to the respective credit rating. Learning algorithms such as decision tree learning algorithm, Support Vector Machines, Neural Network, k–NN, Naive Bayes, Generalized Linear Models and Support Vector Machine (using linear and Gaussian kernel functions), GP and GA have been used as classifiers in various classification tasks [20, 54, 57].

2.2.1 Unsupervised Learning

In unsupervised learning, the data instances have no distinction of types based on input and output variables [50]. Since there is no known output, unsupervised learning excludes training. The goal, therefore, is to discover intrinsic relations or group data instances based on naturally existing affinities between attributes [50]. Clustering and association discovery are examples of unsupervised learning.

2.2.2 Supervised Learning

In supervised learning, features (attributes) are distinctively categorized into dependent (output) and independent (input) variables [50]. The dependent variables (class variables) are to be predicted from the independent variables by training the prediction algorithm (or classifier) on the data instances. Regression and classification are the two main categorizations of supervised learning. In regression there are continuous–valued output to be predicted whilst in classification, the output values to be predicted are discrete.

2.3 Feature Selection and Feature Extraction

In [9], a feature is defined as "an independent measurable property of a process been observed"Noisy attributes in data increase the complexity of the learning space and reduce performance of learning algorithms with corresponding high computational requirement in data analysis and machine learning. Feature selection is the process of refining input data by removing irrelevant and/or redundant features[50]. Feature selection deals with selection of a subset of relevant features (variables) from an input data. Feature selection does not produce new features – all selected features are subsets of the original feature vector.

For a data set D containing m features, a feature selection algorithm selects a subset of n features by eliminating irrelevant and redundant features such that it improves or retains the prediction accuracy of the classifier algorithm. In [15], feature selection problem is defined as: given the original feature vector of Y (Equation 2.1) and a training set L, invent a representation X (see Equation 2.2) derived from Y that maximizes some criterion $J(X)$ (see equation 2.3) and is at least as good as Y with respect to that criterion.

$$Y = \{y_i | i = 1, ..., D\} \tag{2.1}$$

$$X = \{x_i | i = 1, ..., d; x_i \in Y\}_{d \leq D} \tag{2.2}$$

$$J(X^{opt}) = max_{X \subseteq Y, |X|=d} J(X) \tag{2.3}$$

The criterion $J(X)$ is a model or classifier which is used to measure the predictive accuracy of the selected feature subset X. Feature selection algorithms are compared based on performance criteria such as simplicity, stability, number of reduced features, classification accuracy, information gain, storage and computational requirements. In

[9, 20] a number of feature–selection algorithms have been compared on some high dimensional benchmark datasets covering various application areas such as biology, medicine, finance, law, bioinformatics and engineering.

Feature extraction involves the discovery of interesting hidden relationships between original feature vectors. The newly constructed features are linear or non–linear combinations of existing features. The newly formed composite features are fewer, clearer, easy to visualize, more representative of the problem and much more efficient for model extraction. As an example, when estimating the cost of life insurance, an insurance company will like to factor vulnerability of a person to some major disease factors. Attributes such as the height or weight of an applicant could be readily available to the insurance firm. However those attributes in themselves do not necessarily measure any major health risk. On the other hand, an implicit derivation such as the weight to height ratio is crucial to some health risk factors such as obesity –making the new composite feature more valuable than the original features. Even though this attribute will not be explicitly available, the formulation of such key relations between existing attributes is more meaningful to the insurance company for potential health risks.

EAs can be used for the automatic construction of linear/non–linear combinations of features to form a smaller subset of features with improved predictability. GP construct new features by using its tree structure encoding of terminal and function sets to form a linear or non–linear relationship between the original or other composite features. When using GP for feature extraction, the value at the root node of an evaluated GP expression is considered as a new feature. This feature is an implicit derivation from the original feature vectors. Thus, an evaluated GP tree reveals a complex or simple relationship between existing feature vectors making the GP tree a composite feature. The newly created composite features can be used separately or combined with the primitive attributes (original feature vectors) to form a new feature vector.This unique property of GP makes it a suitable algorithm for data mining problems.

2.3.1 Filter Method

In the filter method (see Algorithm 1), feature search method and feature subset selection criterion are independent of the learning algorithm used in the final construction of the classifier [50]. This preprocessing technique ranks features before applying the classifier [or predictor] to the highly ranked features. Ranking methods

are applied to an original feature vector to score feature values based on relevance, which is then used to filter out redundant, irrelevant, and noisy attributes. A criterion is used to determine the rank of a feature using a suitable ranking criterion to select those that meet a predetermined threshold. As an example, a credit firm will want to know key characteristics (or attributes) of clients that determine their credit worthiness. Attribute importance is used to rank each independent feature to produce a value that measures how strong (predictive significance) an attribute contributes towards discriminating the credit worthiness (class label) of a client. In determining feature relevance a unique feature must contain relevant information about the different classes in the data [9]. According to [9, 44], features that have no influence on the prediction of the class labels are irrelevant and can be discarded. One way of obtaining feature ranking is by measuring inter–feature correlation. Degree of correlated features provide some information on inter–feature dependencies. According to [9], "when two or more features are strongly correlated, one is enough to describe the group and the dependent feature provides no extra information for discriminating the class labels". A good feature can be independent of the remaining features but must provide high discrimination for the various class labels. Noisy and redundant features can degrade accuracy of the learning models leading to poor generalization or over-fitting. Over–fitting is a case where the learning model performs well on the training data but poorly on test data. In [9, 52] information gain, minimum description length, maximum relevance, feature-correlation and mutual information are listed as other ways to rank features.

Algorithm 1 Filter Method (from [51])

```
1: procedure FILTER()
2:     Y(F_0, F_1, ..., F_{n-1}) ▷ data set D containing n features
3:     S_0 ▷ initial subset
4:     X ▷ invent a representation X that maximizes some criterion J(X)
5:     S_best ← S_0 ▷ initialize S_best
6:     μ_best ← apply_heuristic(S_0, Y, J) ▷ apply J to S_0
7:     while (!TerminationCriteria()) do
8:         S ← create_subset(Y) ▷ apply a heuristic to create subset
9:         μ ← apply_heuristic(S,X,J) ▷ gets classification accuracy of S
10:        if μ.betterThan(μ_best) then
11:            μ_best ← μ
12:            S_best ← S
13:        end if
14:    end while
15:    μ ← classification_accuracy(S,X,J) ▷ gets classification accuracy of S
16:    return S_best
17: end procedure
```

2.3.2 Wrapper Method

The wrapper method (see Algorithm 2) uses the same learning algorithm used in constructing the final classifier to evaluate feature subsets. The classifier is "wrapped around"a feature–subset search algorithm to determine the subset with the highest predictability to maximize the performance of the classifier.

In a large dimensional data, as N grows exhaustive evaluation of all possible feature subsets become NP–hard. With an exponential growth in feature subsets, exhaustive direct search algorithms such as branch and bound require high computational time and are often intractable as compared to suboptimal search algorithms. Examples of non-exhaustive algorithms applied in the generation of feature subset are the sequential selection algorithms and heuristic search algorithms such as GA. The feature generation process is followed by an external learning algorithm that measures the performance of a selected subset.

Algorithm 2 Wrapper Method (from [51])

```
1: procedure WRAPPER()
2:     Y(F_0, F_1, ..., F_{n-1}) ▷ data set D containing n features
3:     S_0 ▷ initial subset
4:     X ▷ invent a representation X that maximizes some criterion J(X)
5:     S_best ← S_0 ▷ initialize S_best
6:     μ_best ← classification_accuracy(S_0, Y, J) ▷ apply mining algorithm J to S_0
7:     while (!AccuracyImproved()) do
8:         S ← create_subset(Y) ▷ apply a heuristic to create subset
9:         μ ← classification_accuracy(S,X,J) ▷ gets classification accuracy of S
10:        if μ.betterThan(μ_best) then
11:            μ_best ← μ
12:            S_best ← S
13:        end if
14:    end while
15:    return S_best
16: end procedure
```

Sequential selection algorithms start with an empty or full feature set and iteratively add or remove features until a feature subset with maximum classification accuracy is obtained. In the sequential forward selection (SFS) [9] algorithm, the subset generation begins with an empty list, and iteratively adds one feature at a time while choosing the next best feature that yields higher prediction accuracy of the class labels. Sequential backward selection (SBS) starts with a full list of the feature vector and iteratively removes features while accepting the elimination that yields the list change in prediction accuracy of the remaining feature subset. The interested reader is directed to [9] for other variants of SFS and SBS, such as sequential floating forward selection, adaptive sequential forward floating selection, adaptive sequential backward floating selection, and plus-L-minus-R.

In some related works, GA was used for feature generation and decision tree[**?**], k-nearest neighbor (KNN)[37] classifier. [53, 39] used simulated annealing algorithm and a regression based evaluation function for feature subset selection. Alba et all [2] used a hybrid particle swarm optimization (PSO) and GA feature generation and a support vector machine (SVM) classifier. The wrapper method is reported to have better results than the filter approach even though its prone to over-fitting and high computational requirements [9].

2.4 Embedded Method

The embedded method[6] integrates the feature generation strategy and the classification of the quality of the selected subset into the learning algorithm, thus the link established between feature selection and classification is stronger than in the wrapper method [22]. This leads to a combination of the advantages of filter and wrapper based methods with a reduced computational time [9, 22]. The implicit feature generation built into learning algorithms such as decision tree induction algorithms makes CART, ID3 and C4.5 embedded learning methods[41]. By extension, the GP representation based on decision tree induction algorithm also qualifies as an embedded method.

2.5 Evolutionary Algorithms

Heuristic search algorithms include sub–optimal algorithms such as genetic algorithms, genetic programming, particle swarm optimization, ant algorithms, and simulated annealing. The taxonomy of feature selection algorithms is shown in Figure 2.1. Evolutionary Algorithms such as GA and GP borrow principles of natural selection and survival of the fittest from Darwinian theory of evolution to breed a population of feasible individuals using genetic operations such as crossover and mutation. EAs perform parallel evaluation of possible solution sets in the search space by using an objective function that drives the entire search process towards the global optimum. From Algorithm 3, suboptimal population based EAs have the following general characteristics:

(a) The use of a population of individuals (see line 2 of Algorithm 3), which represent a complete or partial solution of the optimization problem. Each solution is defined to meet some predetermined constraints.

(b) A keep–alive evolution process ((see line $4 - 8$ of Algorithm 3)) that breeds new offsprings to replace existing parents until termination criteria is reached. New offsprings are created using genetic operators like crossover and mutation. Crossover swaps a sub–part of the genetic materials from two parents to create two offsprings and mutation randomly alters a sub–part of a parent to create one offspring.

(c) EAs breed a population of solutions and perform parallel evaluation of the solution sets in its search space by using an objective function that drives the entire

search process towards a global optimum. Fitness based individual measurement criteria that assign a value to an individual based on how best it answers the problem. Fitness is used as a criterion for probabilistic selection of parent chromosomes that take part in breeding. The fitness-based selection ensures that most highly fit (strong) individuals pass their genetic material to the next generation.

By using genetic evolutionary operators, and a measure of fitness, EAs are able to perform exploitation and exploration through the search space and drive the entire search process towards promising regions of the fitness landscape. EAs are effective meta–heuristics for searching through large search spaces [14, 16]. A stopping criterion is made on the assumption that individuals are always valid and satisfy all problem constraints. Individuals are bred and evaluated until the EA system satisfies any of these stopping criteria:

(a) The ideal individual has been discovered. This is the individual that correctly classifies all training instances.

(b) The EA system is stuck in local optima such that continuous evolution does not move the entire population from mediocre local optima.

(c) The number of generations specified for a run has been completely exhausted.

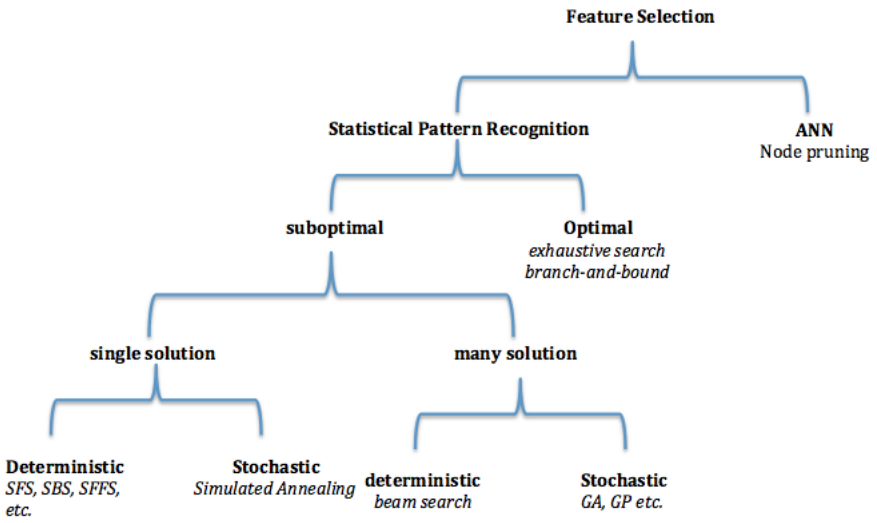


Figure 2.1: A taxonomy of feature selection algorithms [35]

A replacement strategy is used to determine how a new population is produced to replace an old population. The two most popular replacement strategies used in

breeding new population from the existing population are generational and steady state replacements. In generational replacement, individuals in a population are subjected to genetic operations to produce offspring for each subsequent generation. The newly bread population replace the parent population. This new population may include elite individuals that were copied from the old population. The new population of individuals is used to produce the next generation until a termination criteria is met. In a steady state replacement, an offspring individual competes with parents using survival of the fittest or evolutionary survival strategy. The most often used individual replacement strategy is the reverse tournament where an offspring replaces the worst tournament individual.

Algorithm 3 Simple GA

```
1: procedure GEN_GA()
2:     Population ← CreateInitialPopulation ▷ randomly
3:     Generation ← 0
4:     while !( FoundIdealIndividual(j) | TimeIsUp() | MAX(Generation) ) do
5:         EvaluateFitness(Population) ▷ Each individual in Population
6:         Population ← BreedNewPopulation()
7:         Generation++
8:     end while
9: end procedure
```

```
1: procedure BreedNewPopulation()
2:     NewPopulation ▷ initialize new population
3:     i ← 0
4:     SelectionOperation ← {Mutation, Crossover, Reproduction}
5:     for i < popSize do ▷ popSize is user specified
6:         GeneticOperation ← ProbabilisticalSelect(SelectionOperation)
7:         if GeneticOperation = Reproduction then
8:             j ← SelectOneIndividualBasedOnFitness(Population)
9:             k = Reproduction(j)
10:        end if
11:        if GeneticOperation = Crossover then
12:            j ← SelectTwoIndividualBasedOnFitness(Population)
13:            k = Crossover(j)
14:        end if
15:        if GeneticOperation = Mutation then
16:            j ← SelectOneIndividualBasedOnFitness(Population)
17:            k = Mutation(j)
18:        end if
19:        i += k.size()
20:        NewPopulation ← k
21:    end for
22:    return NewPopulation
23: end procedure
```

2.5.1 Genetic Programming

GP [38] is a special case of GA that uses a tree data structure to represent an individual. GP is very flexible and can be adapted to represent various complex patterns from a wide range of domain knowledge. GP was initially introduced with the intention of evolving computer programs. The subject area has expanded to include evolution of mathematical expressions and rule based systems[20]. This expressive power makes it possible to develop dual representations for feature selection and classifier design. In this work, GP is exploited for feature selection and classification by evolving individuals with non–terminal symbols using functions and operations and terminal symbols using the original feature vector. A GP individual is a complete or partial solution represented using a parse tree where constants and variables denote terminal symbols and operators and functions denote non–terminal symbols. Leave nodes correspond to terminal symbols and internal nodes correspond to non–terminal symbols. By the above definition, a function set is the set of all allowed non-terminal symbols and terminal set is the set of all allowed terminal symbols. GPs flexibility can be extended to the construction of various classifiers with representational formalisms such as decision trees, classification rules, discriminant functions, artificial neural networks and many more [20]. The automatic feature selection performed during GPs evolution results in an individual with higher prediction accuracy and contributes to better interpretability of resulting solutions. GP also has the ability to perform automatic feature extraction using a linear or non linear combination of features as sub–trees in a GP individual. A GP language must satisfy the following two conditions:

(a) Sufficiency: this means the GP language including terminal and function sets are enough to evolve a solution for the target problem.

(b) Closure: this is loosely defined as a case where each non-terminal function is able to operate error–free on all values parsed as an input. However in a strongly typed GP language, non–terminals only operate on values that are of the same type as the type of the child node(s). For strongly typed problems, more computational time is used during tree initialization and genetic operations to guarantee type check consistency of nodes in the parse tree.

The specific representation adopted in this thesis is the binary tree based individual representation. A set of individuals make up a population (P) such that $P = \{I_1, I_2, \ldots I_n\}$. An individual I_k is represented by a binary tree (see Figure 2.2),

which is translated into the lisp s–expression

$$(\% \; (x \; (\sqrt{} \; x2) \; x5)(\; (- \; (\% \, (+ \; x5 \; x1)(\sqrt{} \; (- \; x3 \; x6))) \, (+ \; x6 \; x2)))$$

For feature selection problems, terminals are the problem features and are usually combined with randomly generated constants such as an ephemeral constant (ERC). Function sets are operands (such as $*$, $+$, $-$, $\sqrt{}$ in Figure 2.2), which perform operations on the terminals and results of other non–terminals. Tree sizes usually increase in depth as evolution progresses. When two individuals are of the same fitness, the smaller is preferred so as to evolve simple and portable solutions.

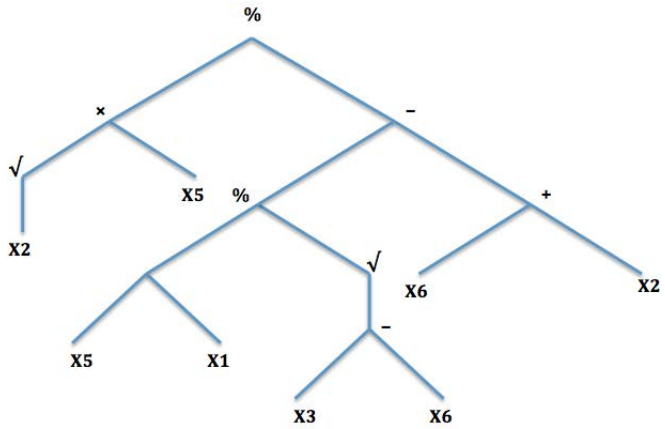


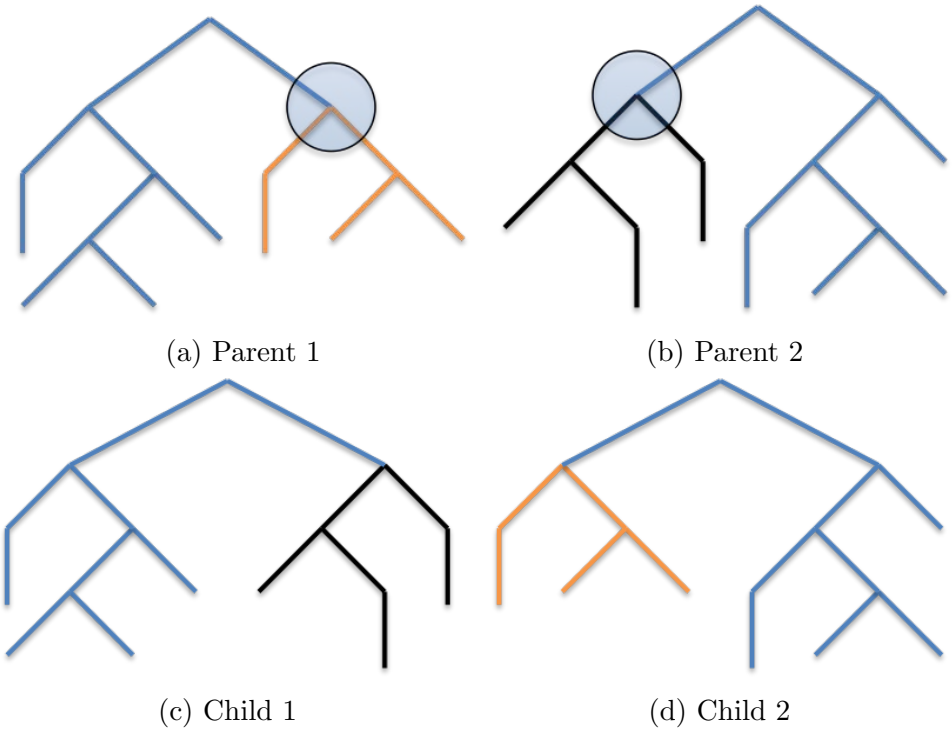
Figure 2.2: GP Individual

The representational power of GP allows implicit feature selection from the terminals of the GP tree while the entire tree can be evaluated to represent an extracted feature. In Figure 4.2, the feature subset is the collection of unique terminal symbols in the set $\{x1, x2, x3, x5, x6\}$. On the other hand, the root value of the GP expression could be evaluated as a new feature, which becomes an implicit derivation from the original feature vectors.

Crossover Operation

During crossover (see line $11-14$ of Algorithm 3), two parent individuals are selected from the population using fitness-proportional selection. A random crossover point is selected from both parents and the sub trees rooted at those nodes are swapped between parents. A typical crossover operation breeds two new offspring as seen in Figure 2.3c and Figure 2.3d from two parent individuals Figure 2.3a and Figure 2.3b. Crossover operations result in exploration of new parts of the fitness landscape.

This is due to the likely combination of genomes from different parts of the fitness landscape.



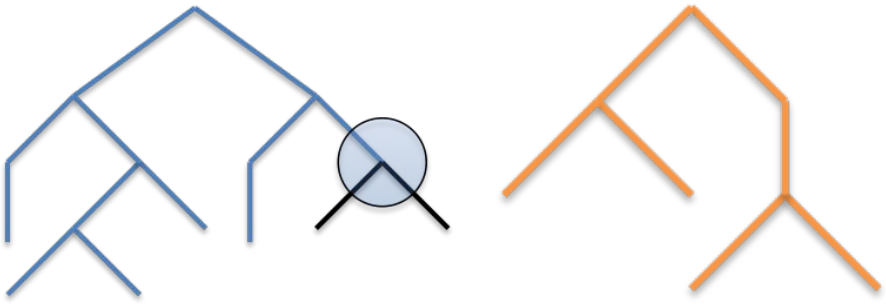
(a) Parent 1 (b) Parent 2

(c) Child 1 (d) Child 2

Figure 2.3: GP Crossover operation

Mutation Operation

In GP, mutation is the process of randomly generating a tree and rooting it on a random node in a parent GP tree (see line $15 - 18$ of Algorithm 3). The parent tree is selected through tournament selection using fitness/random–based selection. In Figure 2.4a, a random node is selected and the randomly constructed tree (see Figure 2.4b) is rooted on the parent individual to form an offspring individual as seen in Figure 2.4c. Mutation is exploitative as it seeks to refine fitness within a particular region of the fitness landscape by rearranging the genes.



(a) Parent with randomly selected mutation point

(b) Randomly generated sub-tree

(c) Parent rooted with randomly generated sub-tree (Figure 2.4b)

Figure 2.4: GP Mutation Operation

2.5.2 Age Layered Population Structure

The Age Layered Population Structure (ALPS) introduced by Hornby[29] is a novel meta-heuristic that seeks to reduce the problem of premature convergence in stochastic algorithms. In ALPS, age is used as a property of individuals to restrict competition and breeding in the population. ALPS segregates the population of individuals into age-layers. Age is measured by how long an individual's genotypic material has been evolving in the population. Hornby[29, 30] applied the ALPS strategy in various problem instances and recorded results that outperformed canonical EA.

An aging scheme (see line 2 in Algorithm 4) is used to separate individuals into age layers (see Table 2.1). These values are multiplied by an age gap parameter to determine the maximum age per layer. Given an exponential aging scheme with an age gap of 10 and 6 layers, the maximum ages for the layers will be $10, 20, 40, 80, 160, 320$. Individuals within a layer are not allowed to outgrow the maximum allowed age

Algorithm 4 Pseudocode for ALPS

```
1: procedure ALPS_GEN()
2:     AgeScheme ← SelectAgeingScheme()
3:     layers ←CreateLayers(AgeScheme)
4:     i ← SequentialLayerSelection(layers).
5: init:
6:     if BottomLayer(i) & reinitializationMode then
7:         j ← CreateNewRandomGenome().
8:     else
9:         if BottomLayer(i) & TooOld(i) then
10:            reinitializationMode ← true
11:            j ← CreateNewRandomGenome().
12:        else
13:            reinitializationMode ← false
14:            j ← CreateNewRandomGenome().
15:        end if
16:    end if
17: stop:
18:    if !( FoundIdealIndividual(j) | TimeIsUp()) then
19:        return false
20:    end if
21: loop:
22:    goto init.
23:    childId ← SelectSlotNextGeneration(i)
24:    j ← CreateChild(childId).
25:    EvaluateChild(j)
26:    TryMoveUp(i,j )
27:    goto stop.
28: end procedure
```

```
1: procedure TryMoveUp(i, j)
2:     if TopLayer(i) then
3:         k ← FindVictim(i,j).
4:         TryMoveUp(i,k)
5:     else
6:         k ← FindVictim(i+1,j)
7:         TryMoveUp(i+1,k)
8:     if Empty(k) then
9:         i ← j
10:    else
11:        Discard(k).
12: end procedure
```

for that layer. Rather, an attempt is made to move such individuals to the next higher layer (see line 26 in Algorithm 4). Unsuccessful migrants are destroyed. The maximum age per layer allows evolution to proceed long enough in the preceding layer before individuals are old enough to migrate to the next available layer. It also allows individuals to improve in fitness before being pushed to the next higher layer. The last layer on the other hand, has no age limit, hence allows for the accumulation of the best individuals. According to [29], an individual in the last layer is only guaranteed to remain there provided it is the global optimum or else it will be replaced by other highly fit individuals from the lower layers. An ALPS system is characterized by the following:

(a) At initialization, individuals are assigned an age of zero (see lines $6-8$ in Algorithm 4).

(b) At regular intervals (determined by the age–gap) new individuals are initialized in the bottom layer (see lines $6-8$ in Algorithm 4). This is when current individuals in layer 0 might have aged into layer 1.

(c) The age of an Individual selected for genetic operation is incremented once per generation for which the individual was used as a parent.

(d) Offspring of parent individuals are assigned the age of the oldest parent plus one.

(e) Breeding is only allowed between adjacent layers using a selection pressure. When breeding, a parent is selected with a probability of $n\%$ from the current layer and $(100-n)\%$ from the lower adjacent layer (e.g. in layer 1, parents are selected from layer 1 and 0, in layer 2, parents are selected from layer 2 and layer 1. This novel process is used to control breeding and competition between individuals in the layers and facilitate the transfer of genotypic materials from lower layer individuals to the higher layers.

(f) Replacement (see line 26 in Algorithm 4) occurs within the population of the active layer. The breeding process enables individuals to age smoothly through higher layers.

Table 2.1: Examples of Age scheme for ALPS [29]

Aging-scheme	0	1	2	3	4	5	6	8
Linear	1	2	3	4	5	6	7	8
Fibonacci(n)	1	2	3	5	8	13	21	34
Polynomial(n^2)	1	2	4	9	16	25	49	64
Exponential(2^n)	1	2	4	8	16	32	64	128

Each successive layer is opened when evolution has occurred for as long as the maximum allowed age for the preceding layer. Thus with the above aging scheme, layer 1 will be opened for evolution at the end of generation 10, layer 2 is opened at the end of generation 20, and so on. The layered approach of evolution does not only restrict competition between individuals in the entire population but also serves as a way of transferring genotypic materials from different fitness basins to higher layers. In ALPS, the bottom layer is regularly replaced with randomly generated individuals. The periodic introduction of such individuals in the bottom layer results in an EA that is never completely converged [29]. By using age to restrict breeding, it reduces the possibility of highly fit old individuals dominating the evolution process, which most often leads to early convergence in canonical EA. Random number seeds are varied for each initialization of base population, which often results in the new population starting off from a new fitness basin. Through the explorative and exploitative nature of genetic operations, breeding could produce an offspring individual that takes the entire population from mediocre local optimum. Each bottom layer restart in ALPS creates a population clustered around a different fitness basin [29], thus the resultant ALPS population increases exploration of the fitness landscape.

When an individual reaches its maximum allowed age in a layer, it will move to the next higher layer. Since a constant population is maintained in all layers, an individual in the next higher layer will have to be displaced to make room for a new individual. In the generational replacement strategy, individual aging in a layer and inter–layer individual movements are synchronized on the condition that when applying selection pressure to pick parents from two layers, a parent selected from a lower layer is not aged. However in steady state replacement, various replacement strategies are adopted for inter–layer replacement. The new individual arriving from the next lower layer is only allowed to displace the weakest tournament individual in the higher layer otherwise it is destroyed. Different replacement strategies such

as replacing population weakest, reverse tournament, random, nearest fitness, etc., could be adapted to the ALPS system inter–layer individual migration.

Chapter 3

Related Work

3.1 Dimensionality Reduction

Meta-heuristics and approximate search algorithms provide suboptimal solutions to NP complete problems using minimal computational resources when compared to direct search algorithms[7]. Suboptimal algorithms have been used for dimensionality reduction on varied benchmark datasets with good classification accuracy and reduced computational requirements[9, 20]. In [20], a methodology is proposed that uses GP to perform feature selection and test the resultant features on a GP constructed classifier. The approach involves construction of a classifier that has n trees for an $n–class$ problem. Dash and Liu [14] use a fitness measure to increase classification accuracy of k–NN classifier while reducing the number of selected features. The accuracy of the k–NN classifier is compared to other statistical methods of feature extraction such as linear discriminant analysis (LDA) and principal component analysis (PCA) [70].

Smith et al. [71] use GP and GA for feature extraction and feature selection respectively as a preprocessor for a $C4.5$ decision tree–learning algorithm. An individual is represented using automatically defined functions (ADFs) such that each individual is made up of separate trees [71]. An ADF is constructed from multiple features and a tree in an ADF is evaluated to produce a newly constructed feature. A GA system is used to evolve a feature subset from the new feature vectors by selecting from the extracted and original features. The resultant feature subset from the hybrid EA system is tested using a $C4.5$ classification algorithm. The approach proved more successful than direct application of the standard $C4.5$ decision tree–learning algorithm on the same dataset. Similar benefits were recorded when the dual preprocessing strategies were performed before using k–NN and Naïve Bayes classifiers. [58], [40] and [18] used $C4.5$ decision tree learning algorithm as a classifier for a GP

feature construction problem.

Oechsle et al. [56] separate feature selection and feature extraction stages and used GP to evolve them independently on a vision problem. The GP classification methodology involved evolution of a set of partial solutions for a single class. This enables the use of GP for classification of high multi–class data and is reported to be faster than conventional GP. In [40] ECJ GP is used for feature construction and feature selection using C4.5 decision tree classifier implemented in WEKA [25]. Lin et al.[49] used a multi–population layered GP to perform feature selection. Two methods of feature selection are proposed in their work. The first method seeks to penalize individuals with high number of features by reducing the fitness values of such individuals. The second method assigns weight (based on feature relevance) to features during evolution; the assigned weights determine survivability of features during evolution. In the implementation of their layered GP [49], best individuals from each sub–population in a layer are considered as an extracted feature. Thus the number of extracted features is limited to the number of sub–populations considered in a layer. These sub–populations have no communication between them. New features extracted from lower layers are combined with existing features to form training set for the next higher layer.

The performance of some feature selection and classification algorithms were compared in [9, 20]. This far, no clear algorithm has emerged dominant for all feature selection problems and results often differ based on considered dataset. Piotr et al. [23] compared feature selection algorithms used in the analysis of chemical data. The results found random forest and support vector machine with recursive feature elimination as superior to other considered strategies. The reader is referred to [9], where a number of feature selection and classification algorithms are compared based on prediction accuracy and reduced feature set.

Badran et al.[3] used a multi-objective GP to perform feature extraction as a preprocessing stage to a multi-class problem. This involved the reduction of the original dimensionality space to a new reduced and optimized multi-dimensional decision space. Thus, Bedran et al. [3] and Lemczyk et al.[45] used GP to handle a multi-class dataset by addressing the classification problem through task decomposition.

The layered GP considered in ALPS restricts communication between some layers. In [49] terminals always have positive weights and the original features are always considered with the extracted features when performing generational weight assignment. We do not penalize individuals with many features as done in [49] but rather, in the FSALPS GP system, we allow natural evolution of feature ranks, such that

irrelevant features gradually disappear under evolutionary pressure.

3.2 Classification

In supervised learning, classification [54] involves the use of an inductive learning method to predict class labels based on information about other features. The goal is to develop a rule (classifier) that assigns each feature vector to a discrete class label. In [9], a classifier is a model encoding a set of criteria that allows a data instance (feature vector) to be assigned to a particular output (class label) depending on some selected features from the data instance. The pre–classified data is divided into training and test set. The classifier is trained on the training example to maximize the number of output scores predicted (correctly categorized) using the input feature vector. The classifier with the highest score of class labels is tested on unseen data (testing example) for its generalizability. A good performance on test example also shows that there was no over–training. Commonly used classifiers are the decision tree–learning algorithm, C4.5[54], k–nearest neighbor algorithm [54], Naive Bayes [46] and other statistical methods such as linear discriminant analysis (LDA)[27] and Principal component analysis (PCA)[70]. Support vector machines (SVM) represents the various samples as points in space. According to [23] these points are optimally separated by hyper–planes such that unique classes have their samples widely separated as much as possible.Given two or more group of samples, LDA [27] finds linear relations between features by maximizing inter–group variance whilst minimizing intra–group variance. Although LDA is simple to implement and often produce good results when the input features are linearly separable, it often does not produce good results on multi–classification problems [23]. PCA is a widely used feature extraction technique derived from linear algebra. It uses a non–parametric method for dimensionality reduction by extracting interesting relationships between existing feature vectors. The interested reader is referred to [70] for a detailed discussion on PCA. In [9, 20], some criterion used for classifier evaluation include:

(a) Classification Accuracy. It is a measure of the number of correctly classified training examples against incorrectly classified training examples.

(b) Generality: ability of selected classifier to score high on unseen data.

(c) Others such as novelty, utility, interpretability etc.

GP can evolve classifiers for binary and multi-class problems. Lemczyk et al.[45] decomposed a multi-class problem using GP and a coevolutionary strategy to evolve classifiers through task decomposition. Their strategy proved very effective on three high-dimensional multi-class datasets.

3.3 Hyperspectral Classification Using GP

GP has the capacity to handle large dimensional datasets. For instance in [43], Langdon used GP to address a large attribute space by reducing the number of features required to describe a cancer diagnosis dataset from a million to 5. Evolutionary computation has many uses in classification, dimensionality reduction and spectral band sub-set selection on hyperspectral datasets. We will investigate the performance (dimensional reduction and classifier construction) of canonical GP, ALPS and Feature Selection ALPS (FSALPS) on a high dimensional hyperspectral dataset. Yin et al.[79] used a modified evolutionary technique —immune clonal strategy to perform band selection of hyperspectral image obtained from Washington DC Mall and Northwest Tippecanoe County. Bazi et al.[4] and Zhou et al.[80] applied a genetic optimization framework for automatic feature selection and parameter optimization for an SVM classifier on hyperspectral remote sensing images. In a related work, Li et al.[47] used the hybrid strategy GA–SVM with a branch and bound algorithm (in the post–processing phase) on a hyperspectral image. [81] used the wrapper based feature selection strategy with GA and SVM classifier to reduce the number of spectral bands of a HYPERION hyper spectral image from 198 to 13. The GA system simultaneously optimized the SVM kernel parameters, resulting in an improvement in the classification accuracy of the reduced spectral bands. Other proposed dimensionality reduction algorithms used for feature reduction and classification of hyperspectral image are the GA based local–fisher's discriminant analysis proposed by Cui et al.[12], GA based feature mining techniques for feature selection and feature extraction[36], and the use of artificial neural networks on remote sensing data[1].

Alex dos Stantos et al.[17] used a content–based image retrieval technique that uses GP to obtain composite descriptors from remote sensing images with an optimum–path forest classifier. Their technique was successfully used in the identification of crop regions on remote sensing images. Due to GPs adaptive learning technique, canonical GP performed a dual function of automatic evolution of classifiers and feature reduction using spectral imagery[62]. Ross et al.[65] applied GP to the Cuprite hyperspectral image for the automatic evolution of mineral identifiers.

In their system[65] canonical GP doubled as a feature reduction and band reduction strategy. The successful and consistent evolution of efficient mineral identifiers for each of the three minerals (alunite, kaolinite and buddingtonite) in the Cuprite area proves the ability of canonical GP to perform feature reduction on high dimensional data sets. Serpico et al.[68], Bazi et al.[5], Guo et al.[24] and Li et al.[47] used varying hybrid strategies to perform classification on the Indian Pines hyperspectral dataset.

3.4 ALPS

The ALPS strategy (see 2.5.2) is a diversity–enhancing algorithm that works with algorithms with elements of randomness in them [29]. It uses an age–layered population and restricts breeding and competition between individuals. ALPS ability to maintain diversity in its population is largely due to regular introduction of individuals from different fitness basins and the novel control of competition between individuals

Layered EA systems such as hierarchical fair competition (HFC) [31], adaptive hierarchical fair competition [32] and continuous hierarchical fair competition [33] usually group individuals into layers based on fitness values [29]. New randomly generated individuals are regularly introduced into the bottom layer and competition is restricted between newly created individuals and high fitness pre existing individuals as is done in fitness uniform selection [34]. According to [29] HFC has a major limitation since newly introduced individuals from different basin of attraction cannot climb through existing high fitness layers. Hornby in [29] found significant performance difference between ALPS, HFC and simple (canonical or standard) EA on an antenna design problem. The reader is referred to [29, 30] where ALPS was explained to be a much effective strategy than other diversity enhancement strategies such as high mutation rate, large population sizes and genotypic diversity strategies etc. Slany [69] compared a classical Cartesian genetic programming (CGP) and ALPS enhanced CGP on an image operator evolution problem. In all six test cases compared, the ALPS enhanced CGP outperformed the classical CGP. Patel et al. [59] tested and compared ALPS GP and standard GP on the evolution of non–linear factor models for financial portfolio optimization. The work in [59] also compared reduction in premature convergence, over–fitting of GP solutions to training data and the overall best fitness obtained in both training and testing. A t–test on all test cases proved superiority (significant difference) of the ALPS GP system over standard GP and market index values.

Schmidt et al.[67] used a Pareto formulation of ALPS that uses the age of an

individual and its performance to evolve solutions on a two dimensional pareto front. The proposed approach overcame the problem of premature convergence, quickly discovered good solutions in a reduced computational time on a symbolic regression problem.

In [26] a co–evolution is used with ALPS in an algorithm called "Spatial Co–Evolution in Age Layered Planes (SCALP)" to attempt a bloat reduction strategy in GP. When SCALP was compared to operator equalization on a bivariate regression problem, the hybrid strategy "SCALP"was found to avoid bloat without any integration of an express bloat reduction strategy. When tested on a classification dataset, SCALP produced short, general solution trees as opposed to operator equalization and canonical GP.

Chapter 4

System Design

This thesis uses a new algorithm based on a modification of the ALPS algorithm –feature selection age layered population strategy (FSALPS). FSALPS was designed by making some modifications to the ALPS algorithm. One such area is in inter-layer individual migration strategy. Also, a number of feature ranking methods were developed for calculating probability values for each feature. The essence of testing multiple strategies is to aid in the empirical selection of the most appropriate method for the problem types considered in this work.

4.1 Inter-Layer Migration

The ALPS strategy requires regular inter–layer migration for individuals that are older than the allowed age limit for a layer. An individual's age is generationally compared to the maximum allowed age of its layer, and if it is older, an attempt is made to move the individual to a higher layer. All layers are scanned for the layer (L_h) with an age range that can accommodate a new individual from a lower layer (L_l). A number of replacement strategies were tested when moving over–aged individuals (I_l) to a higher layer. For each of these strategies, either the replacement individual is always replaced or a conditional replacement is made. In the conditional replacement, the replacement individual (I_h) is replaced only if its fitness is worse than I_l. The following replacement strategies were implemented.

4.1.1 Nearest In Population (NIP)

The fitness of all individuals in L_h is compared to fitness of I_l. Individual I_h with the nearest fitness to I_l is picked as a replacement individual (I_h is destroyed) and

replaced by I_l and finally, I_l is removed from the lower layer (L_l). The downside of this strategy is a new individual I_l that is of a similar fitness could replace the best individual in L_h. This replacement strategy could introduce some new healthy genes into the higher layer; however, it could kill other strong individuals in L_h.

4.1.2 Worst In Population (WIP)

The weakest individual (I_h) in population L_h is selected and replaced by I_l, which in turn is removed from the lower layer L_l. Due to the restricted breeding between individuals in the ALPS population, I_l could be coming from a new fitness basin, and may result in a more diversified population with prospects of further improving new fitness peaks. Weak individuals are sometimes very helpful in the evolution process due to some healthy genes that might be present in their chromosome. The "greedy" termination of such individuals in the WIP strategy can lock the evolutionary process in a local optima and result in a population of highly fit individuals. These could likely reduce the effect of genetic operators in breeding strong offspring that have higher fitness than their parents.

4.1.3 Random

A random individual (I_h) is selected from population L_h and marked for replacement. I_l replaces I_h and L_l is destroyed from the lower layer. This inter–layer replacement strategy is chaotic in nature because the best individual in L_h could be destroyed. This approach can be very destructive, especially when moving multiple individuals from a lower to higher layer. Although evolution is subject to a random process (which to some extent could be useful for evolution), the random replacement strategy had a negative effect on elitism. Thus, it was almost impossible to track elite individuals in the population.

4.1.4 Reverse Tournament Nearest (RTN)

RTN is similar to NIP, except that the sample space from which the victim individual is selected is restricted to the tournament individuals. In an extreme sense, it will be equivalent to NIP if the tournament size is the same as the population size. Tournament selection is performed on L_h, k individuals are selected, and a victim individual I_h is marked for replacement. The victim individual has the closest fitness to I_l. Although the new similar fitness individual could drive the entire population

towards discovering new promising regions on the fitness landscape, there is also a chance of destroying strong individuals, including the best individual in L_h.

4.1.5 Reverse Tournament Worst (RTW)

The worst tournament individual (I_h) from L_h is selected for replacement. This replacement strategy guarantees that the best individual in L_h is never replaced thereby preserving the best individual while selectively replacing weak individuals in the population. Tournament selection requires a careful regulation of selective pressure. Since the best tournament individual is not selected for replacement, there is the issue of reverse selective pressure that has to be controlled. A high tournament size usually results in replacement of weak individuals in L_h population. A reasonably small tournament size (empirically determined) on the other hand will be the fairest since other strong individuals could be marked for replacement.

4.2 FSALPS

4.2.1 FSALPS Details

FSALPS is used to speed up the feature selection process by providing a loop back mechanism (see Figure 4.1) where evolved feature-counts are translated into feature–ranking probabilities and are used for selecting terminal symbols for tree construction during initialization or for sub–tree construction during mutation. The FSALPS heuristic requires regular refinement of features in the GP population through integration of a feature count process that is translated into terminal–probability values for the selection of leave nodes. The terminal-probability values are used to guard also serves as feature ranks and are used during tree initialization and mutation.

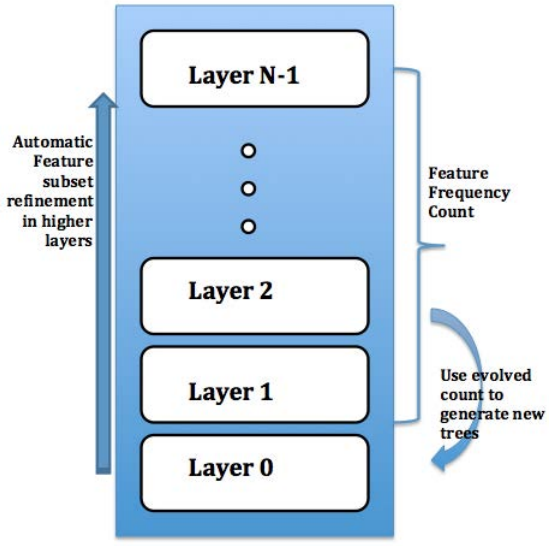


Figure 4.1: GP Individual

The algorithm for FSALPS is shown in Algorithm 5 and its based on a modification of the ALPS algorithm (see Algorithm 4). Lines 2−5 initializes the FSALPS algorithm by loading default probability values for all terminal symbols from a parameter file. Otherwise the system automatically assumes an equal probability for all terminal symbols when there are no such specifications. FSALPS begins a frequency count for all terminal symbols when individuals are migrated into the last layer (or when the last layer becomes active). There is the option to count features in the entire ALPS population or only features in specified layers as shown on line 8 Algorithm 5.

The frequency count values are then converted into probabilities on line 8 of Algorithm 5 by invoking Algorithm 6. These newly assigned probabilities for the features are used in the random creation of new individuals when the FSALPS system enters re–initialization (see line 9 and 13) mode otherwise it is used in mutation (see line 16) for the random selection of terminal symbols for the construction of a sub-tree to be rooted on a parent node.

Algorithm 5 Pseudocode for FSALPS

```
1: procedure FSALPS_GEN()
2:     AgeScheme ← SelectAgeingScheme()
3:     layers ← CreateLayers(AgeScheme)
4:     i ← SequentialLayerSelection(layers)
5:     probVector ← InitialFeatureProbabilities()
6: init:
7:     if BottomLayer(i) & reinitializationMode then
8:         probVector ← ComputeFeatureProbabilities()
9:         j ← CreateNewRandomGenome(probVector)
10:    else
11:        if BottomLayer(i) & TooOld(i) then
12:            reinitializationMode ← true
13:            j ← CreateNewRandomGenome(probVector)
14:        else
15:            if mutation then
16:                j ← CreateNewRandomGenome(probVector)
17:            else if crossover
18:                j ← CreateNewRandomGenome()
19:            end if
20:        end if
21:    end if
22: stop:
23:    if !( FoundIdealIndividual(j) | TimeIsUp()) then
24:        return false
25:    end if
26: loop:
27:    goto init.
28:    offspringIndex ← SelectSlotNextGeneration(i)
29:    j ← CreateChild(offspringIndex)
30:    EvaluateChild(j)
31:    TryMoveUp(i,j )
32:    goto stop.
33: end procedure
```

Algorithm 6 Pseudocode for Feature Ranking

```
1: procedure ComputeFeatureProbabilities()
2:     FeatureVector ← null
3:     ProbabilityVector ← null
4: loop:
5:     for k in Layer(i) do
6:         for terminal in k do
7:             FeatureVector[terminal]++;
8:         end for
9:     end for
10:    sum ← Sum(FeatureVector)
11:    for t in FeatureVector do
12:        ProbabilityVector ← t/sum
13:    end for
14:    return ProbabilityVector
15: end procedure
```

4.2.2 Feature Analysis

A frequency count of features in the best solution (classifier) was done. This is done by a direct count of features in the solution tree. Given a problem with 9 unique features ($F0 - F9$), the number of subsets that can be constructed is 2^9. The GP expression tree in Figure 4.2 performed automatic feature subset selection from an original feature vector of size 10. Table 4.1 shows feature frequency–count of for the GP individual 4.3. Table 4.1 contains a Boolean column "Count"that indicates if a feature was used in a solution. For each solution, the count of the number of "1"indicates how many features were used in that expression. In the above example, the feature count is 5.

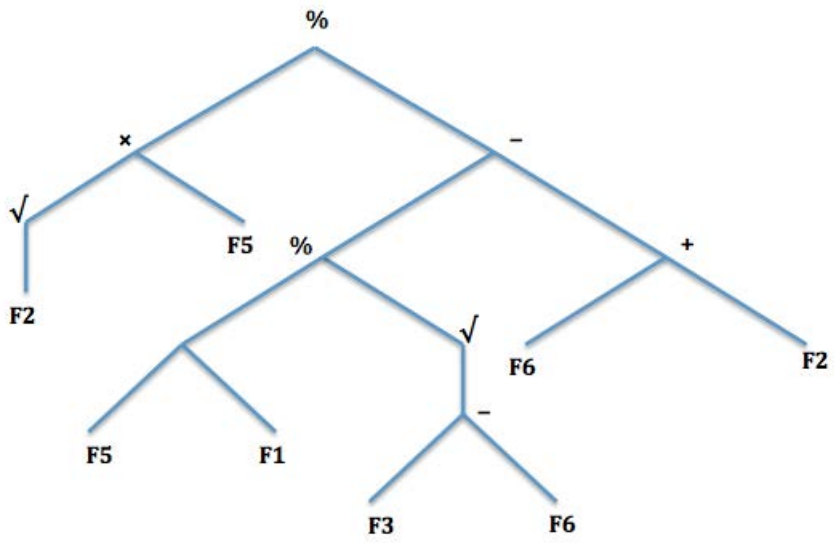


Figure 4.2: GP Individual

Table 4.1: Frequency Calculation Strategies

Feature	Frequency	Count
F0	0	0
F1	1	1
F2	2	1
F3	1	1
F4	0	0
F5	2	1
F6	2	1
F7	0	0
F8	0	0
F9	0	0
Total	8	5

4.3 FSALPS Frequency Calculation Strategies

When the last layer becomes active in the FSALPS system, a probability calculation is initiated using the terminal frequency count. The probability values calculated from the frequency count are used in terminal selection during mutation and during initialization of the first layer. A number of probability calculation strategies were

developed and tested on the FSALPS system. In each of these strategies, there is the option to either use node count of only the last layer, or all layers. Layer 0 is usually noisy since not many genetic operations would have been applied to individuals before aged to Layer 1. In the first initialization of the bottom layer, all terminals are given the same probability or the parameter–specified probability values until the automated frequency calculation is started. The noisy nature of Layer 0 does not significantly affect probability distribution when the entire population in all layers is used in computing probability values. The probability values of terminals are directly proportional to the chance of selecting a node during tree construction, thus a node with a high probability has a high chance of selection. Also, relying on frequency count from only the highest layer could be too drastic and lead to premature convergence. The probability calculation strategies that were tested are as follows:

4.3.1 Normal Frequency

This requires direct conversion of terminal frequency distribution into terminal probabilities (see Equation 4.1). Two terminal sets (x and y) with a frequency count of 20 and 5 respectively will produce probability values of 0.8 and 0.2 respectively. As shown in Table 4.2, the normal frequency calculation performs feature elimination drastically since all terminal symbols with a zero count are assigned a zero probability. Given the drastic elimination of features, this strategy could limit exploration of the search space and might result in discovery of mediocre solutions.

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{4.1}$$

4.3.2 Uniform Frequency

Uniform frequency calculation requires the addition of the average frequency of all terminals to the frequency count of each terminal (see column "Uniform Freq" in Table 4.2). Given 5 terminals, and an active last layer, uniform frequency is calculated as shown in column "Uniform Prob" of Table 4.2, using Equation 4.2. This gives a chance to terminals that could have otherwise disappeared when using Normal Frequency and is quiet beneficial during the early stages of evolution in the last layer. This also reduces the chance of the GP system getting stack in local optima, especially when the extinction of a feature could prevent the discovery of an ideal individual with high generalization of class labels.

$$P_i = \frac{f_i + \frac{1}{n} * \sum_{j=1}^{n} f_j}{\sum_{j=1}^{n} (f_j + \frac{1}{n} * \sum_{k=1}^{n} f_k)} \tag{4.2}$$

4.3.3 Ranking Frequency

Ranking frequency uses frequency counts to rank all terminals (see column "Rank" in Table 4.2). Terminals with larger counts have bigger ranks. The ranks are then converted into probability values using Equation 4.3 (see column "Rank Prob " in Table 4.2). Once again, probability changes when using this scheme is not as drastic as the normal frequency and often gives a chance to terminals that are almost getting extinct from the GP system.

$$P_i = \frac{rank_i}{\sum_{j=1}^{n} rank_j} \tag{4.3}$$

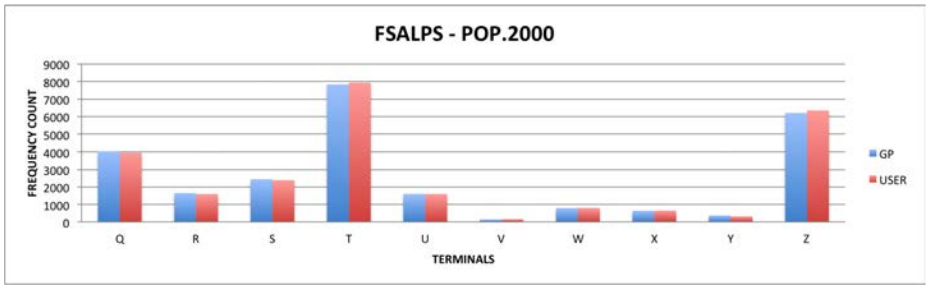
Table 4.2: Frequency Calculation Strategies

Instance	Normal Freq	Uniform Freq	Rank	Normal Prob	Uniform Prob	Rank Prob
A	20	40	3	0.20	0.20	0.214
B	50	70	5	0.50	0.35	0.357
C	20	40	3	0.20	0.20	0.214
D	10	30	2	0.10	0.15	0.143
E	0	20	1	0.00	0.10	0.071

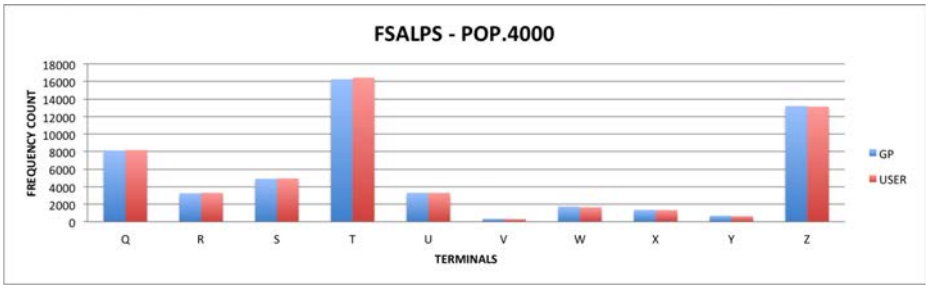
4.4 Tree Initialization

Tree initialization in GP involves random construction of individuals using terminal and non–terminal symbols. FSALPS controls the selection of terminal symbols based on evolved terminal–probability values. As a test, the GP system in ECJ[66] was modified to allow specification of direct probability values for the nodes. A 10–feature problem was used to measure how effective the probability system worked towards the selection of leave–nodes for constructing tree expressions. The parameter–specified probability values were compared to node–count of terminal symbols after random

initialization of individuals. Results in Figure 4.3 show an accurate translation of these parameter–specified probability values into random initialization of GP trees. As expected, higher population sizes resulted in closer ties between the specified and expected values.



(a) Population 2000



(b) Population 4000

Figure 4.3: Comparing parameter specified terminal probabilities with terminal selection during tree initialization. Accuracy is affected by population size, higher population produces more accurate results

4.5 Fitness Function

For each feature subset, a classifier is used to test the prediction accuracy of class labels. This evolution mechanism is directed towards refining feature subsets and evolving a classifier with high discrimination of class labels. The fitness function converts the number of correctly predicted class labels into classification accuracy. Fitness is maximized; meaning strong individuals have high fitness values.

In a binary classification problem, the fitness of individuals on a test data with P positive instances and N negative instances is translated into metrics such as true positive (TP), true negative (TN), false positive (FP) or type I error and false negative (FN) or type II error. Class labels that were correctly discriminated by the evolved GP classifier make up TP and TN. True class labels that were correctly identified are TP while false class labels that were identified as such by GP are TNs. On the other

hand, true class labels that were incorrectly identified by the GP system as false are FN while false class labels that were classified as true by the GP system are FPs. The sum of TP and FN equals the sum of positive class labels in the test data. In a classification problem, the four metrics discussed can be used to measure sensitivity (true positive rate) and specificity (true negative rate) of the classifier. From the work of [49] Sensitivity or true positive rate is given in Equation 4.4 and specificity or true negative rate in Equation 4.5.

Other fitness evaluation criteria [77] that could be derived are positive predictive value (precision), negative predictive value, false positive rate (fall out) and false negative rate (miss rate). Accuracy is calculated using Equation 4.6

$$sensitivity = TP/P = TP/(TP + FN) \tag{4.4}$$

$$specificity = TN/N = TN/(TN + FP) \tag{4.5}$$

$$accuracy = (TP + TN)/(P + N) \tag{4.6}$$

$$fitness = sensitivity * specificity \tag{4.7}$$

4.6 Effect of Bloat on Frequencies

Bloat is a condition in GP where an increase in tree size does not lead to an increase in fitness. The bloat expressions do not contribute to the fitness of the individual except it increases the size of the individual without a corresponding increase in evaluation time. The frequency count values in FSALPS are translated into feature ranks therefore it is best to avoid count of terminal nodes in bloat expressions. It is difficult to completely eliminate bloat from a GP solution tree, however the following precautions were taken to reduce the occurrence of bloat.

The GP language used for the experimentations in this work is less likely to result in bloat. The language excludes conditional non–terminal symbols that easily produce bloat expressions. In Figure 4.4, the huge bloat sub–tree does not contribute to fitness of the individual and must be avoided during frequency count. A bloat scanner could be used to isolate bloat expressions however it will shoot up evaluation time and make the algorithm less efficient.

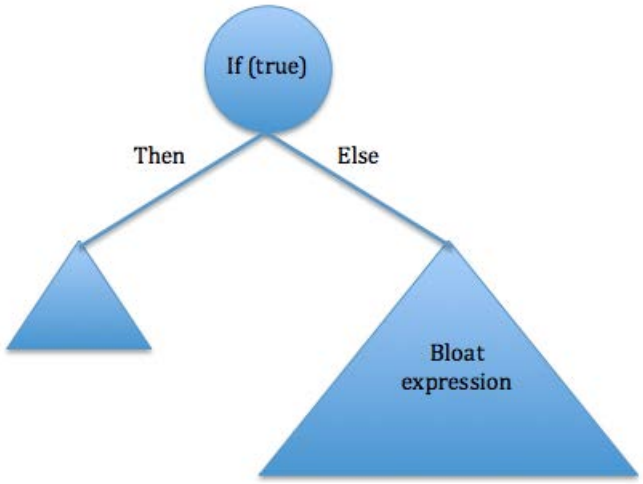


Figure 4.4: Bloat expression in a GP individual

A tree depth restriction was enforced to ensure that tree expressions do not grow beyond a specified depth. The tree depth is a Koza [38] rule used to restrict the growth of trees. If a crossover produces an offspring with depth larger than 17, its smaller parent replaces the offspring. Individual size is also made a factor when performing selection operation. A smaller tournament individual with an equal fitness to a larger tournament individual is always preferred. However there was no limit to the maximum number of nodes that could be used in the construction of an individual.

Ideally if bloat was detected, it would be helpful because bloat trees adversely affect frequency counts in FSALPS. Bloat therefore leads to inaccurate information about features and affects feature relevance.

4.7 Hardware Configurationn

Five sets of experiments were investigated in this thesis. We setup the Pima Indians diabetes, Breastcancer, Ionosphere and Sonar classification experiments on a matrix cluster with 9 nodes with 6 cores each making a total of 54 cores with 2 different hardware configurations:

1. Intel Core i7 920 2.66GHz with 12GB ram
2. AMD Phenom X6 1055 3.6GHz with 8GB ram

The runs were deployed in parallel such that each run was handled by one compute node.

The Corn–notill and Soybean–mintil classes of the Indian Pines hyperspectral experiment were deployed on 16 nodes each having 4 cores, making up a total of 64 cores and a hardware configuration of Intel Core i5 2500 3.3GHz with 4GB ram per node.

Chapter 5

Experiments

In this chapter, we first look at empirical determination of some experimental parameters. We introduce the datasets used for testing canonical, ALPS and FSALPS. A detailed experiment is conducted to determine how GP, ALPS and FSALPS perform on feature selection and classification tasks using these datasets.

5.1 Datasets

Four datasets with varying number of features and training examples were used to measure feature selection and classification performance of the three algorithms investigated. Pima Indians diabetes, breast cancer, Ionosphere and sonar datasets are continuous (floating point) values obtained from the UCI [48] repository. These data sets can be used in a number of different ways to test learning speed, classification accuracy, quality of ultimate learning, ability to generalize, or combinations of these factors. Each of them is a binary classification problem so we evolve a classifier to discriminate the two class labels. The accuracy of a classifier is measured by how many labels are correctly predicted in a set of cases.

One common way of overcoming over–fitting in classification problems is to randomly reorder the training examples before applying the predictor algorithm [71] Song et al. [74] used subset selection with genetic programming to control over–fitting. This approach ensures that the evolved classifier is not over fit to a particular training subset and usually leads to an improved test results for large data sets. We will now follow up with a discussion of the classification datasets used in this work.

5.1.1 Pima Indians Diabetes

The Pima Indians diabetes dataset is obtained from the UCI[48] repository and has 8 attributes and 768 instances. All candidates considered for this dataset are females with a minimum age of 21 and of Pima Indian heritage. The attributes are:

(a) Number of times pregnant

(b) Plasma glucose concentration a 2 hours in an oral glucose tolerance test

(c) Diastolic blood pressure ($mmHg$)

(d) Triceps skin fold thickness (mm)

(e) 2-Hour serum insulin (muU/ml)

(f) Body mass index ($weight\ in\ kg/(height\ in\ m)^2$)

(g) Diabetes pedigree function

(h) Age ($years$)

(i) Class variable (0 or 1)

The dataset was cleaned to remove all instances with missing data. This is to enable the learning algorithm to construct high–accuracy learning models that are valid and applicable to to real test data. After cleaning, there were a total of 336 records with 225 instances of negative cases and 115 positive cases.

5.1.2 Breast Cancer Wisconsin (Diagnostic) Data Set

The breast cancer Wisconsin (Diagnostic) dataset [48] has 32 real valued attributes and 569 instances. According to [48] the features are computed from a digitized image of a fine needle aspirate of a breast mass that describes characteristics of the cell nuclei present in the image [48]. The attributes are described as follows:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. [3–32] Ten real-valued features are computed for each cell nucleus:
    (a) radius (mean of distances from center to points on the perimeter)

(b) texture (standard deviation of gray-scale values)

(c) perimeter

(d) area

(e) smoothness (local variation in radius lengths)

(f) compactness ($perimeter^2/area$–1.0)

(g) concavity (severity of concave portions of the contour)

(h) concave points (number of concave portions of the contour)

(i) symmetry

(j) fractal dimension ("coastline approximation"- 1)

The dataset has no missing values so all 569 instances were used for training and testing. We design a binary classifier to predict the class labels (malignant or benign) of the breast cancer data. The number of attributes supplied to the learning system is 30 and each individual in the GP system is trained using all data instances assigned for training.

5.1.3 Ionosphere

The ionosphere data set consists of 351 data instances and 34 attributes per instance. According to [48] all 34 attributes are continuous and made of a phased array of 16 high–frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The class labels to be predicted are either good or bad; where "Good" radar returns are those showing evidence of some type of structure in the ionosphere and "Bad" returns are those that do not [48]. The dataset is used for binary classification and has not missing values. The first 34 attributes are continuous and the 35^{th} attribute is either "Good" or "Bad".

5.1.4 Sonar

Attributes for the sonar dataset were obtained by bouncing sonar signals off a metal cylinder or rock at various angles in varying conditions [48]. There are 208 instances and 60 real valued features for each data instance within the range $0.0 - 1.0$. According to [48] 111 of the data instances were produced from a metal cylinder and the remaining 97 from rocks under similar conditions. There are no records of mission values for the sonar dataset.

5.2 Parameter Tuning

The GP parameters used were empirically determined. A non-exhaustive parameter sweep was conducted by using variations of parameters on feature selection and classification problems. A broad range of parameters such as elitism, tournament size, population size, number of evaluations, ageing schemes, ALPS replacement strategies, and FSALPS frequency calculations strategies were varied to determine acceptable performance for canonical, ALPS and FSALPS.

Parameter tuning is done to find the best parameter values most favorable to each of these EA strategies. We followed up by comparing all three strategies using parameter settings with promising performance for each algorithm. For instance, when performing $250,000$ evaluations for canonical, do we use a population size of 250 and 1000 generations, or a population size of 2500 and 100 generations, or population size of 25 and $10,000$ generations. These considerations are balanced between all three algorithms to ensure that one algorithm is not given an unfair chance to outperform the others.

5.2.1 Tournament Size

Tournament size was varied between 3, 4 and 7 to determine the effect of tournament size on evolution. A high selective pressure usually results in the selection of strong individuals for parents and is usually the case when the tournament size is high. A good selective pressure allows the selection of some weak individuals as parents during breeding especially because some of these weak individuals contain some strong traits that must be preserved. Chances are that those good genes could help move the population out of a mediocre fitness peak leading to the discovery of new improved solutions. In [29] a tournament size of 7 was used, in other cases such as [9, 63] lower tournament sizes were used. We compared these tournament sizes on a Pima Indians Diabetes classification problem; the results did not yield any significant difference. This is probably because these values were neither too low nor too high and had a balanced selective pressure.

5.2.2 Number of Evaluations

Another parameter that often requires a good balance is the number of observed evaluations. This often determines how long an EA runs. In generational canonical GP, the number of evaluations is the product of the population size and the number of gen-

erations per run. Smaller evaluations do not allow evolution to progress long enough for the discovery of better solutions. On the other hand, long evaluations are not always helpful due to the likely introduction of overtraining. Overtraining is caused by the EA memorizing training values and usually results in poor generalization of test data. Overtraining must be avoided especially for classification problems since a performance measuring criterion for constructed models include good generalization on unseen data.

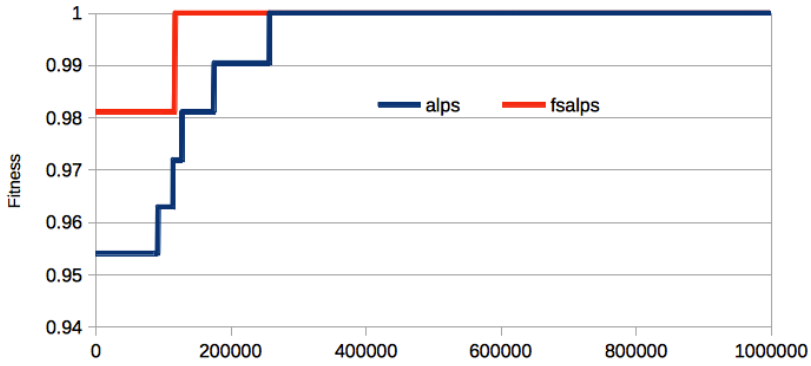


Figure 5.1: ALPS and FSALPS overtraining on Sonar dataset

A typical example of an overtraining is seen in Figure 5.1 where for 1 million evaluations, ALPS and FSALPS scored 100% on training data however the resultant classifier generalized poorly on a test data with ALPS and FSALPS scoring best results of 55.76% and 59.6% respectively then average scores of 45.81% and 41.88% respectively over 20 runs.

5.2.3 Population Size

In a population based stochastic algorithm, the size of the population is a major factor in determining the extent to which the fitness landscape is explored. Also, a large population affects initialization and evaluation time (for breeding the next generation). Therefore, one key consideration in selecting parameters for an EA is to determine the right population while considering the capacity of the hardware on which the experiment will be performed. Although the search process is random, smaller population sizes often converge on local optima solutions thereby making it difficult to discover promising areas on the fitness landscape. The population size used for all subsequent experiments were empirically determined.

5.2.4 ALPS Parameters

The ALPS strategy was implemented in ECJ [66] to work with GA and GP evolutionary algorithms. The system was designed to support two replacement strategies; steady state and generational. The wide range of various strategies accommodated in the design was to allow for empirical determination of some key evolutionary parameters. Migration of over–aged individuals from a lower ALPS layer to a higher layer could be accomplished using different strategies. In this work, strategies such as NIP, WIP, random, RTN and RTW were implemented but RTW was used as a preferred strategy. See Section 4.1 for a discussion on these strategies.

5 layers were used in each ALPS system using the polynomial (1, 2, 4, 9 and 16) ageing scheme. The maximum age for a layer is obtained by multiplying the ageing scheme by the age-gap size. In this case maximum age for layer 0 is 5, layer 1 is 10, layer 2 is 20 etc. When selecting parent(s) for breeding, the selection pressure parameter determines how often a parent is selected from either the current or immediate lower layer. A selection pressure of 0.8 implies 80% of parents are selected from current layer and 20% from the immediate lower layer. See Appendix A.1 for a discussion on how parameters such as replacement strategy and ageing scheme were empirically determined.

5.3 Parameters

The final evolutionary parameters used to set up canonical GP are specified in Table 5.1. Additional parameter settings used in the FSALPS and ALPS systems are listed in Table 5.2. The interested reader is referred to [38] for discussion of some basic GP parameters. Each evolutionary run will complete a total of $250,000$ evaluations. Each dataset is shuffled and divided into 20 subsets (k–fold size is 20). We then iteratively use each subset for training and the remaining $(k-1)$ subsets for testing.

Table 5.1: Canonical GP Parameters

Parameter	Definition
Number of observed Runs	20
Replacement Strategy	Generational
Population size	250
Generations	1000
Selection	Tournament selection = 4
Initial Population	ramped half-and-half
Grow Minimum	2
Grow Maximum	6
Maximum tree size	17
Crossover	90 %
Mutation	10 %
Termination criteria	100% classification accuracy or end run
k-fold cross validation size	20

Table 5.2: ALPS and FSALPS parameter settings

Parameter	Definition
Number of observed Runs	20
Number of Layers	5
Offspring Selection pressure	0.8
Population per Layer	50
Elite size per layer	3
Aging scheme	Polynomial (1, 2, 4, 9, 16, 25, 49, ...)
Age gap size	5
Layer Replacement	Reverse Tournament Worst
FSALPS Probability Calculation	Normal Frequency

5.4 GP Language

The function sets listed in Table 5.3 were used to operate on the terminal symbols. All function sets were defined to operate error free on expected attributes. As an example the green node (div function) in Figure 5.2 will evaluate to 1 if the second argument F0 is 0. It is important the functions are defined to meet closure requirements to avoid illegal operations during tree evaluation.

Table 5.3: GP Language

Name	Representation	Arity	Definition
Log base 10	$\log 10$	1	$\begin{cases} log10(arg0), & arg0 \geqq 0 \\ 1, & arg0 < 0 \end{cases}$
Natural Log	ln	1	$\begin{cases} ln(arg0), & arg0 \geqq 0 \\ 1, & arg0 < 0 \end{cases}$
Maximum	max	2	$Maximum(arg0, arg1)$
Minimum	min	2	$Minimum(arg0, arg1)$
Multiplication	*	2	$arg0 * arg1$
Addition	+	2	$arg0 + arg1$
Subtraction	-	2	$arg0 - arg1$
Protected Division	%	2	$\begin{cases} arg1/arg0, & arg0 \neq 0 \\ 1, & arg0 = 0 \end{cases}$
Features	$f_0 - f_n$	0	Datasets (see section 5.1)

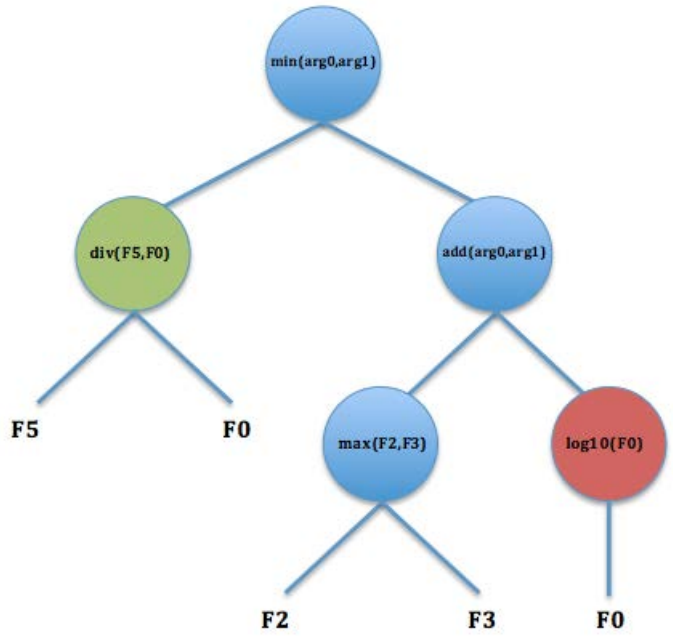


Figure 5.2: Evaluation of GP tree expression

5.5 Cross Validation

In classification problems, a predictive model (classifier) is trained on a known data set (training data) and the best performing classifier with highest classification/prediction accuracy is tested on new testing data. Predictive models are tested for how accurately they are able to classify class labels from datasets that were not used for training. Testing is essential because the constructed classifier is now tested on a new unseen dataset. This is ideal for measuring how the classifier overcomes over–fitting and generalizes independent test data.

Cross validation partitions a given dataset into training and test subsets. The test subset is the validation set used in measuring performance of the model mostly in real–world problems. Cross validation will be very useful in guarding against type I errors (or"false positives"), type II errors (or "false negatives") and reducing type III errors [55].

Exhaustive cross validation is when learning and testing is performed by considering all possible variations of sample data into training and testing data. The non-exhaustive cross validation does not exhaustively compute all possible combinations of input data into training and testing samples. A number of variations exist for the non–exhaustive cross validation [76].

This technique is very popular for testing classification accuracy of a classifier (model). The original data sample is split into m sub–samples. k–fold cross–validation (as described by Wu [78]) for a single run of k–fold cross–validation proceeds as follows:

1. Randomly reorder the data sample
2. Partition the original sample into k subsamples (or folds) where each subsample has approximately m/k examples.
3. (a) For i=1,. . . ,k:
   (b) Train the classifier on all examples that do not belong to sample i
   (c) Test the classifier on sample i
   (d) Calculate wrongly classified examples k_i
4. The classifier error ε (see Equation 5.1) is calculated using

$$\varepsilon = \frac{\Sigma_{i=1}^{k} n_i}{m} \tag{5.1}$$

The accuracy for k–fold cross validation could be improved by running the single k–cross validation t times. From [78] let ε_1,. . . ,ε_t be the accuracy estimates obtained for all t runs. The estimate for the algorithm performance is the error (e in Equation 5.2) as seen in Equation 5.3 with standard deviation $\sigma = \sqrt{V}$

$$e = \frac{\Sigma_{j=1}^{t} \varepsilon_j}{t} \tag{5.2}$$

$$V = \frac{\Sigma_{j=1}^{t} (\varepsilon_j – e)^2}{t-1} \tag{5.3}$$

5.6 Comparing ALPS, FSALPS & Canonical

This work compares the performance of ALPS GP, FSALPS GP and Canonical GP on different classification datasets. All three algorithms are compared using carefully tuned parameters that allow for a fair comparison. In the ALPS and FSALPS experimentation, a new parameter setting was used that differed from what Hornby [29] used when comparing ALPS and Canonical GP. Given that GP is prone to premature convergence; which sometimes leads to mediocre non-improving solution, an

equivalent parameter setting for the canonical GP should be designed to match the effort used in both in ALPS and FSALPS. A slight deviation will be made from the use of elitism in Hornby's comparative parameter settings. In a generational ALPS experiment with n layers each having k elitism per generation, Hornby used an elitism parameter of $n\ x\ k$ for an equivalent canonical experiment.

The high elitism value in the canonical experiment could easily drive the entire population towards an early convergence. This is because most of the highly fit individuals will be similar thereby reducing the effectiveness of genetic operations on new offspring due to reduced diversity. In our approach, the same elitism parameter used per layer in ALPS is used in the canonical setting. By reducing the number of generations, we will prevent the overly long evolution in which fitness converges to local optima. Each run is initialized with a different random number generation seed and likely starts the population from on a new fitness peak.

FSALPS constantly analyzes the population and computes frequency counts that are converted to probabilities. These probabilities are used in the selection of terminal symbols during tree construction. See Appendix A.1 for a complete discussion of feature count and probability calculation.

5.7 Results

To ensure that experimental results are consistent with the tested algorithms, twenty runs were conducted for each experiment. This guarantees that results are not based on a random process and goes to prove consistency in all observed runs for each algorithm. Using k–fold cross validation ($k = 20$), each run is performed using a unique subsample of testing examples while the remaining are used as training examples. The training results for all three algorithms are discussed in Appendix A.4.1. We did not observe signs of premature convergence in all three algorithms. In this section, we will analyze performance of the best training individual on test data.

5.7.1 Classification Accuracy

The feature selection problems investigated in this work were done using GP towards selection of relevant feature subset from a given feature vector. GP solutions trees were evolved to simultaneously perform feature subset selection and classification (see Appendix A.4.1 for a discussion on training performance). The performance (fitness score) of the GP individuals is measured on a training dataset. The prediction

accuracy of a classifier is a measure of how the predictor performs on all training examples. The best solution tree in the training phase is tested on a test dataset. Testing is performed to ensure that the evolved classifier is not over-fitted to the training dataset and that it is generalizable on an unseen data. Using k–fold cross validation, all instances of the datasets are used for testing. Table 5.4 contains the average testing performance (classification accuracy measured in percentage) for each k–subset of the best-evolved training solution tree. The results recorded for all three strategies were not significantly different when compared at a 95% confidence interval using Tukey's HSD ANOVA test. Table 5.5 shows the level of significance scored on the test dataset where an arrow points to the superior strategy and a –shows an insignificant difference between both strategies.

Table 5.4: Prediction accuracy (%) on test data using $250,000$ evaluations

	Canonical		ALPS		FSALPS	
	Average	Best	Average	Best	Average	Best
Breastcancer	93.12	100.0	92.80	100.0	93.82	100.0
Ionosphere	88.02	100.0	89.71	100.0	90.65	100.0
Pima	73.47	94.12	73.81	88.24	74.03	94.12
Sonar	71.27	100.0	73.68	100.0	74.36	100.0

Table 5.5: Comparing mean difference between classification accuracy for each strategy at the 0.05 significance level using $250,000$ evaluations for Breastcancer($\otimes$), Ionosphere($\oplus$), Pima($\ominus$) and Sonar($\oslash$) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

	ALPS	FSALPS
Canonical	—$\oplus\otimes\ominus\oslash$	—$\oplus\otimes\ominus\oslash$
ALPS	—	—$\otimes\oplus\ominus\oslash$
FSALPS		—

5.7.2 Feature Analysis

Feature analysis is computed for best solutions in each of the 20 runs – Table 5.6 summarizes feature reduction per dataset. The summary table also shows aver-

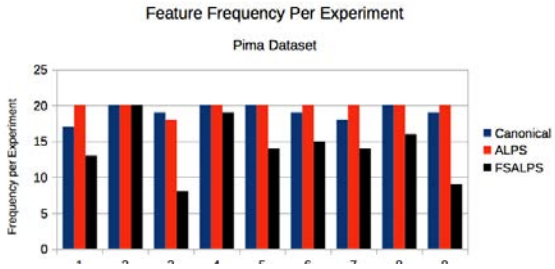
age/min/max number of features within best solution trees per experiment. This summary table immediately shows that overall feature reduction is happening especially in the FSALPS strategy.

Table 5.6: Minimum (Min), maximum(Max) and average (Avg) number of unique features in best solution tree for all runs

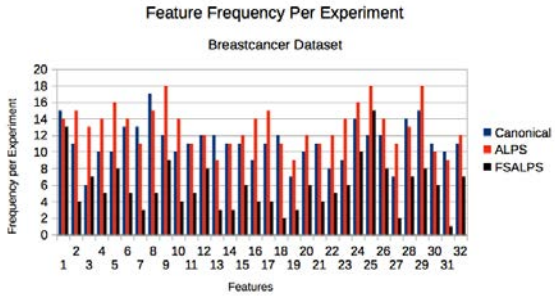
Dataset		Canonical	ALPS	FSALPS
Pima: Total = 9	Min	7	8	3
	Max	9	9	9
	Avg	8.6	8.9	6.4
Breastcancer: Total = 33	Min	12	6	5
	Max	26	29	17
	Avg	17.85	20.9	9.3
Ionosphere: Total = 35	Min	9	13	7
	Max	25	31	20
	Avg	18.65	23.5	12.75
Sonar: Total = 61	Min	16	28	7
	Max	42	45	27
	Avg	23.65	36.9	16.85

Table 5.7: Comparing mean difference between feature selection performed for each strategy at the 0.05 significance level using Tukey's HSD ANOVA test for Breastcancer($\otimes$), Ionosphere($\oplus$), Pima($\ominus$) and Sonar($\oslash$) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

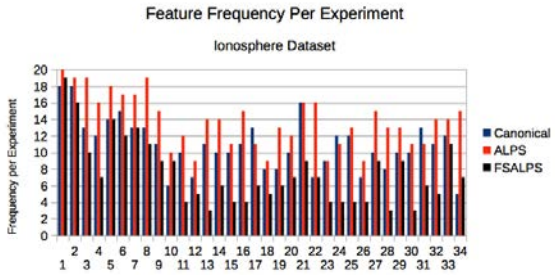
	ALPS	FSALPS
Canonical	$—^{\otimes\ominus}$ $\longleftarrow^{\oplus\oslash}$	$\uparrow^{\otimes\oplus\ominus\oslash}$
ALPS	—	$\uparrow^{\otimes\oplus\ominus\oslash}$
FSALPS		—



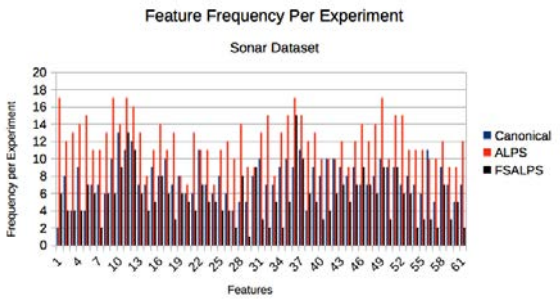
(a) Pima



(b) Breastcancer



(c) Ionosphere



(d) Sonar

Figure 5.3: Histogram of cumulative feature used in solutions per dataset for 20 experiments

The histograms in Figure 5.3 reports the cumulative appearance of each feature for

all 20 runs. The histogram(s) break down the frequency table into the actual features themselves. Average feature–count histograms are biased towards the number of times a feature might be duplicated in the solutions, which is proportional to the tree size. The chart showing presence of a feature in a solution will show the contributions of features within solutions themselves, at a more basic level. (Of course, it does not account for features used in bloat terms that have no effect on fitness). In the Pearson correlation coefficients obtained for all three strategies (see Table 5.8), FSALPS and ALPS report high correlation as opposed to Canonical GP.

Table 5.8: Pearson Correlation Coefficients for Figure 5.3

	Canonical vs ALPS	Canonical vs FSALPS	ALPS vs FSALPS
Pima	−0.183	−0.254	0.914
Breastcancer	0.545	0.587	0.803
Ionosphere	0.624	0.675	0.786
Sonar	0.333	0.391	0.589

Some features were almost completely eliminated for each subsequent run. The feature elimination process is demonstrated especially in the FSALPS strategy. Classification accuracy recorded for canonical, ALPS and FSALPS is not different at 95% confidence interval, we can therefore confidently rely on the superior feature selection of FSALPS. The reader is referred to Appendix A.4.2 for a further discussion on feature evolution in the GP population. We used Tukey's HSD ANOVA test to compare feature selection ability of all three strategies on the various datasets. Table 5.7 proves the superiority of FSALPS as a dominant feature selection algorithm at 95% confidence interval.

5.7.3 Tree Size Analysis

Tree size is measured by counting the number of nodes contained in a solution tree[1]. In Fig 5.4 the average tree growth rate per generation for canonical GP is very high as compared to ALPS and FSALPS. The memory requirement for canonical GP increases at a faster rate as compared to ALPS and FSALPS. ALPS and FSALPS quickly attained constant memory utilization at higher generations. The efficient

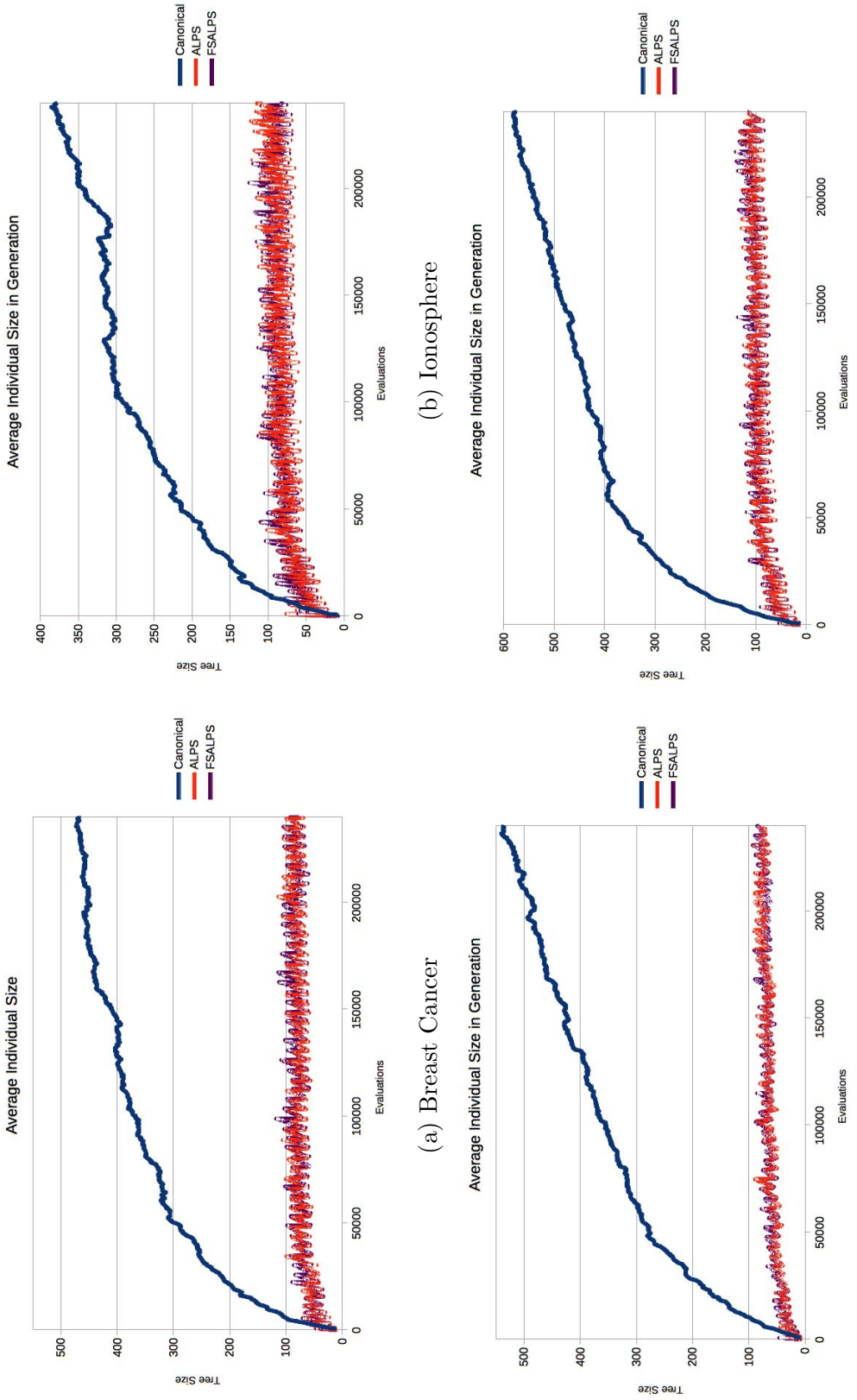
[1]see Appendix A.4.4 for some sample GP trees

memory utilization for ALPS and FSALPS makes them a preferable algorithm for the problems considered in this work.

The tree size of the best solution tree used for testing was compared at a 95% confidence interval using Tukey's HSD test. The results show a significant size difference between the canonical GP and ALPS variants for all but one dataset. Further details on time complexity and memory usage are provided in Appendix A.4.3.

Table 5.9: Comparing mean difference between best solution tree-size for each strategy at the 0.05 significance level for Breastcancer(⊗), Ionosphere(⊕), Pima(⊖) and Sonar(⊘) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

	ALPS	FSALPS
Canonical	—$^{\oplus}$ ↑$^{\otimes\ominus\oslash}$	—$^{\oplus}$ ↑$^{\otimes\ominus\oslash}$
ALPS	—	—$^{\otimes\oplus\ominus\oslash}$
FSALPS		—



(a) Breast Cancer (b) Ionosphere

(c) Pima (d) Sonar

Figure 5.4: Average individual size per generation for 250,000 evaluations

5.7.4 Time Analysis

Two separate time analyses were performed on the population: initialization and evaluation time. Aside from classification accuracy, which is a direct measure of how each algorithm performs on unseen data, the running time for each algorithm determines how fast an algorithm is in providing quick solutions to large problem sets. One key purpose for using evolutionary algorithms for NP complete problems is to overcome the time overhead involved in using direct algorithms on such problems.

5.7.5 Initialization Time

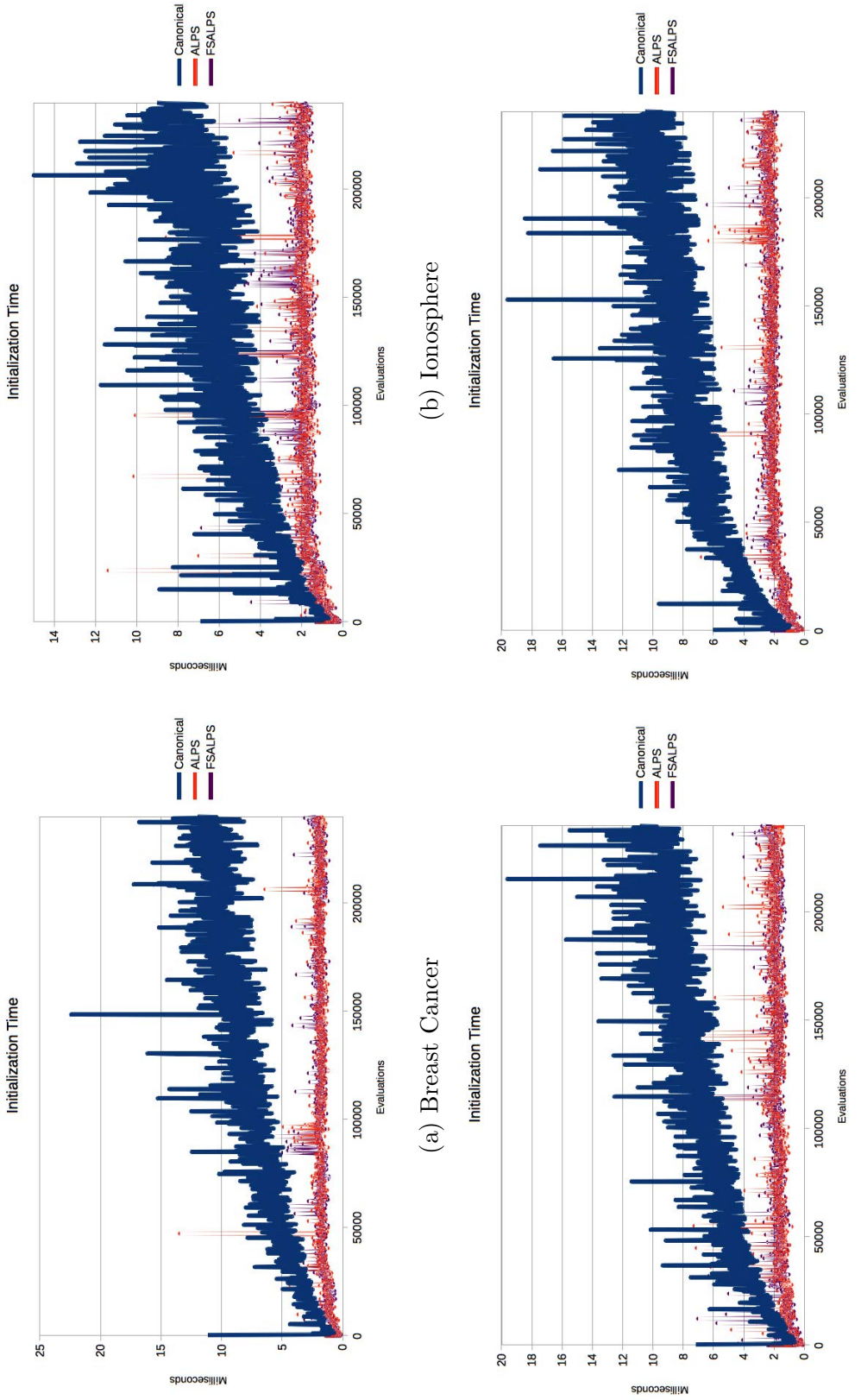
Initialization time is the total time taken to breed the initial population. For subsequent generations, initialization time is how long the previous generation took to breed a new population. Initialization time is usually proportional to the population size and the size of trees used for breeding. The bottom layer in ALPS and FSALPS has a fraction of the population in canonical GP. In ALPS and FSALPS, only the bottom layer is initialized at every initialization; this explains why these two metaheuristics almost always record smaller initialization time as compared to canonical GP. Another aspect of initialization time is the time taken to breed a new population. Yet again, canonical GP is seen to have the highest breed time, which is as a result of the high population bred at a goal as well as possible large size of solution trees in canonical GP. The ALPS variant almost always has less population per layer to breed, due to the constant introduction of new and small but fit individuals from different fitness basins into higher layers, genetic operations often does not result in cross breeding of huge individuals. The tree depth is restricted, thus when the population is converging on large trees (as seen in canonical GP), chances of breeding offspring individuals with larger than allowed depth is high. Such individuals are destroyed and new ones bred in their place. This repeated attempt of breeding offspring individuals with required depth contributes to an increased initialization time at higher generations. The top layer in ALPS and FSALPS contain ancestral individuals and is most likely to breed large individuals. However, the last layer only contains a fraction of the entire population and is responsible for the reduced evaluation time as seen in Figure 5.6.

A key improvement introduced by the ALPS paradigm is its ability to maintain a population of highly fit individuals with small tree sizes. The regular introduction of new and small individuals from lower layers means breeding often involves average size individuals. The use of a selection pressure to control breeding between layers to allow

selection of parent individuals from two layers (with possible smaller individuals in the bottom layer) serves as a control strategy for the size of offspring individuals especially when applying crossover. Average individual size from lower layers is smaller than from upper layers. When breeding is allowed between two such layers with varying individual sizes, offspring individuals are often a hybrid of the two layers and serves as a control mechanism for breeding average sized individuals.

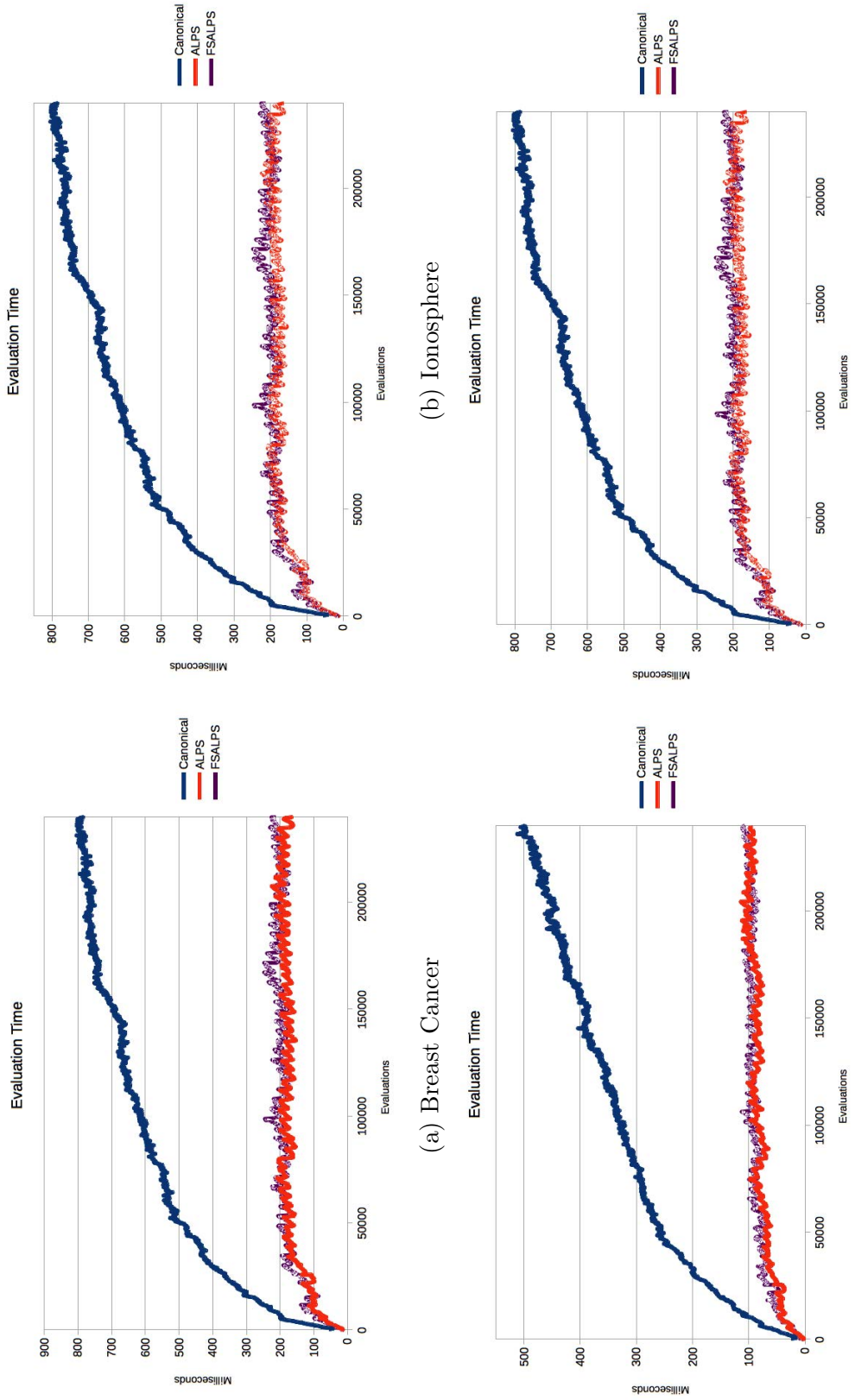
Before breeding, FSALPS does an analysis of frequency count of features in the entire population; these frequency counts are converted to probabilities that are used for selection of terminal symbols for constructing trees/sub–trees. Due to the tree analysis and probability generation performed in FSALPS, it was expected that the breeding time recorded for FSALPS will be higher than ALPS but it is not the case as seen in Figure 5.5. This is only possible because the directed feature selection mechanism taken place in FSALPS also results in the generation of smaller but high-fitness offspring with refined feature sets. Breeding using relevant features often results in the construction of smaller and easily interpretable individuals. The expected reduced initialization time is not seen in Figure 5.5 because of the extra-time used in computing the feature probabilities.

Before breeding, FSALPS does an analysis of frequency count of features in the entire population; these frequency counts are converted to probabilities that are used for selection of terminal symbols for constructing trees/sub-trees. Due to the tree analysis and probability generation performed in FSALPS, it was expected that the breeding time recorded for FSALPS will be higher than ALPS but it is not the case as seen in Figure 5.5. This is only possible because the directed feature selection mechanism taken place in FSALPS also results in the initialization of offspring with refined feature sets. Breeding using relevant features often results in the construction of smaller and easily interpretable individuals. The expected reduced initialization time is not seen in Figure 5.5 because of the extra effort used in computing the feature probabilities. ALPS and FSALPS regularly introduce small trees into the population. These trees age quickly through the entire population and are responsible for the smaller initialization time observed. Canonical on the other hand only produces smaller trees at the start of evolution and tree size grow under evolutionary pressure. This is responsible for the longer initialization time seen in Figure 5.5.



(a) Breast Cancer (b) Ionosphere (c) Pima (d) Sonar

Figure 5.5: Initialization time using 250,000 evaluations



(a) Breast Cancer (b) Ionosphere

(c) Pima (d) Sonar

Figure 5.6: Evaluation time using 250,000 evaluations

5.7.6 Evaluation Time

Evaluation time is the time taken to evaluate an entire GP population. The ALPS variants usually have fewer populations in each layer, as compared to the population evaluated in one canonical GP generation. Evaluation time is influenced by the amount of time used in traversing all nodes in the GP tree expression when calculating fitness. Thus algorithms that produce smaller tree sizes have corresponding smaller evaluation times. Canonical GP usually results in the breeding of larger trees at higher generations as seen in Fig 5.4. The spontaneous growth in the tree size is usually caused by crossover with huge sub trees, which also produces large bloat expressions. The presence of bloat is supported by the fact that the average tree size plot shows a huge margin of growth as compared to the best tree size plot (see Appendix A.4.3). Canonical GP continuously breeds huge trees without any significant improvement in fitness but rather increases the running time of the algorithm. The growth is only limited by the tree depth parameter that prevents offspring from outgrowing a specified depth. ALPS and FSALPS have an almost similar evaluation time, which is due to the unique ability of these two strategies to breed small but strong individuals.

For all datasets considered in this work, the evaluation time for ALPS and FSALPS are significantly smaller than what is recorded for canonical GP (see Figure 5.10). Given that classification accuracy for the evolved classifiers were not significant, it proves ALPS and FSALPS as a better alternative to canonical GP.

Table 5.10: Comparing mean difference between initialization and evaluation time for each algorithm at the 0.05 significance level for Breastcancer($\otimes$), Ionosphere($\oplus$), Pima($\ominus$) and Sonar($\oslash$) dataset for all 20 runs. An arrow points to the dominant strategy while a — means there is no significant difference between the two strategies.

	ALPS	FSALPS
Canonical	$\uparrow^{\otimes\oplus\ominus\oslash}$	$\uparrow^{\otimes\oplus\ominus\oslash}$
ALPS	$\longleftrightarrow$	$—^{\otimes\oplus\ominus\oslash}$
FSALPS		$\longleftrightarrow$

5.8 Discussion

Computational resources are finite which usually dictates the development of efficient algorithms. Suboptimal algorithms are used to produce good solutions to NP

complete problems. The time and space requirement for such algorithms are very important as it measures their feasibility on limited computational resources. Applying direct algorithms on NP complete problems is computationally intractable due to huge time and space requirements. In this work, we have tested the performance of canonical, ALPS and FSALPS GP on feature selection and classification problems. We relied on Tukey's HSD one–way ANOVA test at 95% confidence interval to test for differences in the various algorithms on a number of performance indicators.

We were surprised not to have noticed a significant difference classification accuracy results for all three strategies on the test data. The results however showed a significant difference in feature reduction at the 95% confidence interval. Canonical GP outperformed ALPS in terms of feature reduction. It was the case because of the random re–introduction of new features into ALPS population. ALPS maintain a diverse, never–converging population due to its periodic re–introduction of new individuals into the population. However, this negatively impacts the feature selection capability of ALPS. In ALPS, new trees/sub–trees are constructed by randomly selecting features from the terminal symbols (original feature vector). On the other hand, FSALPS performs a directed feature selection by using evolved feature–counts to dictate selection of new features during tree–initialization and sub–tree mutation operations. FSALPS performs relevant feature subset selection while relying on ALPS search power and overcoming the problem of premature convergence. At the 95% confidence interval, FSALPS outperformed canonical and ALPS GP in selecting the most minimal relevant features to represent the original feature vector. The superiority of FSALPS as a dimensionality reduction strategy over canonical and ALPS proves its usefulness as a pre–processing strategy in selecting relevant subsets of features for data mining problems. By significantly reducing the number of features required to describe a given data, the huge hypothesis space of a large dimensional NP complete problem is significantly reduced. We have shown through the experiments that FSALPS can reduce the curse of dimensionality problem, thus opening up a new frontier for machine learning and data mining problems.

ALPS and FSALPS also showed very interesting results in terms of evolved solution size. When compared with canonical GP, the observed difference in tree size was significant at the 95% confidence level. The ability of the ALPS variants to breed smaller trees proportionally affected the time required for initializing and evaluating trees. This was responsible for the reduced lower run–time of ALPS and FSALPS. The huge cut in computational time and memory requirements for ALPS and FSALPS makes them an efficient alternative to canonical especially for data mining and knowl-

edge discovery tasks.

Chapter 6

Feature Reduction Using Hyperspectral Image

Hyperspectral images are acquired using imaging spectrometers and have applications in areas such as resource management, agriculture, mineral exploration and environmental monitoring. This imaging technique allows for simultaneous collection of image data in dozens of narrow adjacent spectral bands [73] and is achieved using hyperspectral remote sensors, which are usually airborne (mostly on planes or satellites). The study of hyperspectral images offer some benefits over remote sensing usually due to its ability to measure a narrow band of spectra using a wide range of wavelengths. Hyperspectral images are ideal for classifying various landforms characterizations such as vegetation or diagnostic materials associated with ore deposits [73]. This is possible because various organic and inorganic materials offer unique absorption and reflection properties at particular bandwidths. This makes it possible to identify specific objects of interest by carefully studying and comparing the absorption and reflectance properties of remotely acquired images with known properties of some existing spectra. Hyperspectral images contain useful information that could be extracted by understanding relevant ground properties and been able to relate them to measured features on the hyperspectral data. Advancement in hyperspectral images has made it possible to detect natural resources in areas inaccessible by foot and enhanced the understanding of mineral or natural resource distribution across wide areas of landforms. The classical way of mapping these surface materials involves adjustments in atmospheric effects and terrain effects then comparing the resultant image spectra to a field acquired reflectance spectra [73].

The hyperspectral cube usually contains a large number of bands. As an example

the AVIRIS[1] sensor of NASA is able to take 224 different bands in the range of 0.4–2.5μm wavelength whiles the Canadian Aeronautics and Space Institute sensor of Itres Research of Canada is able to take 288 bands in the range of 0.43–0.87μm range [73]. The large number of bandwidths considered per hyperspectral cube increases the difficulty involved in detecting objects. This difficulty is deepened by the presence of noise or spectra signatures from non–interesting objects such as water vapor.

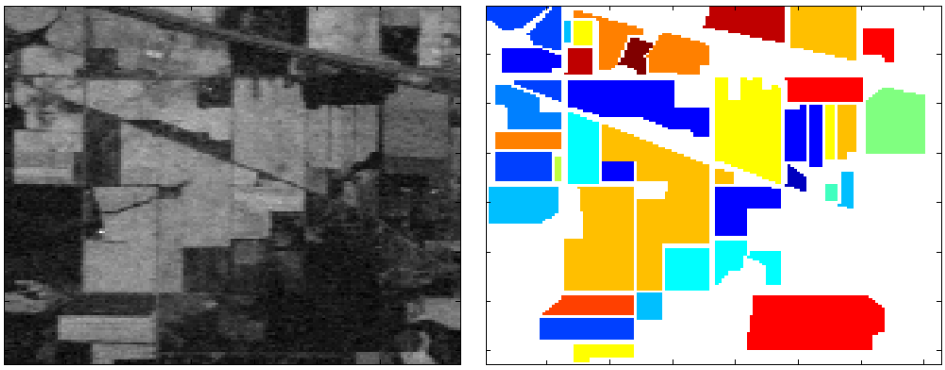
GP has been used in agribusiness for the purpose of precision agriculture. Alex dos Stantos et al.[17] used a content–based image retrieval technique that uses GP to obtain composite descriptors from remote sensing images with an optimum–path forest classifier, for the identification of crop regions. Chion et al.[11] used a hybrid genetic programming and spectral vegetation index strategy to identify regions on hyperspectral data to aid in precision farming. Rauss et al.[62] used GP for the automatic evolution of classifiers to categorize hyperspectral imagery. Mingyi et al.[28] applied a hybrid system – GP and error–correcting output codes for the classification of multi–spectral and hyperspectral data. Brian et al.[65] applied GP to the Cuprite hyperspectral image for the automatic evolution of mineral identifiers. This system leveraged the representational power of GP Canonical GP to perform feature reduction, band reduction, and simultaneous evolution of a classifier.

We will use GP to evolve a classifier for detecting specific class labels (crops) on an Indian pines hyperspectral data [64]. The GP system is trained using manually selected points on each spectra of the hyperspectral image to be detected. We will evolve separate rules that predict the various class labels under investigation. The classifier is able to predict the presence of a class label at a given coordinate on the hyperspectral image when supplied with vector data from a particular pixel on the hyperspectral cube. A fitness function is used to measures the performance of a classifier on positive and negative training examples. The best classifier of the run is tested on all pixel objects on the hyperspectral cube to determine how best it is able to discriminate the class labels from non–interesting objects. We will also compare the performance of canonical GP, ALPS and FSALPS on the Indian pines hyperspectral data. The intention is to determine which of these strategies produce the best performance metrics especially regarding feature selection and classification accuracy when applied to large dimensional datasets.

The Indian Pines hyperspectral dataset is a 145 by 145 pixel data obtained over the Indian Pines test site in North–western Indiana using the AVIRIS sensor[64]. The data contains 224 spectral reflectance bands with a wavelength range of 0.4–2.5μ

[1] Airborne Visible/Infrared Imaging Spectrometer

meters. The Indian Pines site predominantly consists of agriculture (two thirds), and one third forest and other natural perennial vegetation. Other scenes on the site are highways, rail line, housing and other built structures. The original data set acquired from [21] contains 220 bands that were subsequently reduced to 200 bands by [64] by removing bands $104 - 108$, $150 - 163$ and 220.



(a) Sample band of Indian Pines dataset 2.09μm

(b) Groundtruth of Indian Pines dataset

Figure 6.1: A sample spectral band from the Indian pins hyperspectral data and its ground truth image

According to [64] the reduction gets rid of spectral artifacts such as water from the earth's absorption band and solar flux. Fig 6.1a shows an AVIRIS hyperspectral reflectance image of the Indian Pines data at the 2.09μm bandwidth. The resulting surface reflectance of 200 bands contains the spectral information necessary for the identification of the classes of interest. The ground truth image for the Indian Pines data contains 16 classes with known samples (see Fig 6.1b). The classes on the ground truth image are listed below:

(a) Class

(b) Alfalfa

(c) Corn–notill

(d) Corn–mintill

(e) Corn

(f) Grass–pasture

(g) Grass–trees

(h) Grass–pasture–mowed

(i) Hay–windrowed

(j) Oats

(k) Soybean–notill

(l) Soybean–mintill

(m) Soybean–clean

(n) Wheat

(o) Woods

(p) Buildings–Grass–Trees–Drives

(q) Stone–Steel–Towers

6.1 Experimental Setup

In this experiment, we will be evolving an identifier for different crops on the Indian pines dataset. This will require training of a classifier on positive and negative examples obtained from the hyperspectral image. Positive examples are obtained from regions on the Indian pine dataset that are to be identified and negative examples are any other area on the image other than the region of interest. High dimensional datasets increase the difficulty of evolving efficient classifiers therefore we will rely on GPs ability to perform automatic feature selection. For each pixel, we normalize the RGB color channel between 0–1 using $x - min/(max - min)$ where max is the maximum RGB value on the image data and min is the minimum.

Each crop type has strong distribution on particular areas of the hyperspectral data. This means we only have low intensity spectra at the edges of the class samples to be detected hence we do not have to evolve crop identifiers for areas with different thresholds as done in [65] on a mineral identification problem. In this work, no special edge filters are used to identify boundaries of resident spectra with weak reflectance, but rather, we use the normalized pixel values to evolve classifiers for areas with high intensity reflectance of the resident spectra.

GP is given adequate information to evolve crop identifiers for interesting regions on the hyperspectral data. The training examples were carefully selected to include

difficult regions with slightly low threshold of resident spectra. 95% of training examples are from regions with 100% intensity of observable reflectance of class labels. The remaining 5% are boundary regions with reduced intensity of resident spectra. The number of handpicked training examples used for both crops are specified in Table 6.1.

Table 6.1: Number of samples used per training class

Class	**Positive samples**	**Negative samples**
Corn–notill	73	102
Soybean–mintil	60	150

Our choice of GP as an evolutionary system is because of GPs dynamic representational power. We will use GP to evolve crop identifiers for the hyperspectral data. Each GP program in the population is evaluated against all training examples and assigned a fitness score that corresponds to how accurately it predicts the class labels. A terminal symbol is extracted from each band and does include only one spectral property and no spatial filters or moving filters. All 200 terminal symbols extracted from the hyperspectral cube are normalized to floating point values. We also include a floating–point ephemeral constant (ERC) bounded by $-1 <= ERC <= 1$ to help fine tune detection of difficult areas on the spectral data. All complex computations are carried out at program initialization phase[2]. At the start of evaluation, data for terminals are parsed to individuals through array indexing (memory referencing).

The GP system used is the java based evolutionary system ECJ [66]. The system is integrated with ALPS (see Section 3.4) and FSALPS (Section 4.2).

6.1.1 Parameters

The parameter listing in Tables 6.2 and 6.3 were empirically determined and are based on best performance from previous experiments discussed in Section 5.2 of Chapter 5. Table 6.2 contains a self–explanatory parameter listing for canonical GP. ALPS and FSALPS are supplemented with further parameters in Table 6.3. Similar parameters supplied in Table 6.3 override those supplied in Table 6.2.

[2]The program initialization phase is also the setup phase in GP. The parameter file and all other pre–evolution operations are computed.

In FSALPS, the normal frequency count (NFC) parameter is used in counting terminal frequencies in an FSALPS population. The frequency count starts prior to every bottom–layer restart after individuals are shipped to the last layer. All terminal symbols in the FSALPS population are counted and converted to probability values that are used to direct terminal selection during tree/sub-tree construction.

Table 6.2: Canonical GP Parameters

Parameter	**Definition**
Number of observed Runs	20
Replacement Strategy	Generational
Population size	250
Generations	1000
Selection	Tournament selection = 4
Elite size	3
Initial Population	ramped half-and-half
Grow Minimum	2
Grow Maximum	6
Maximum tree size	17
Crossover	90 %
Mutation	10 %
Termination criteria	100% classification accuracy or end run

Table 6.3: ALPS and FSALPS parameter settings

Parameter	**Definition**
Number of observed Runs	20
Number of Layers	5
Offspring Selection pressure	0.8
Population per Layer	50
Elite size per layer	3
Aging scheme	Polynomial (1, 2, 4, 9, 16, 25, 49, . . .)
Age gap size	5
Layer Replacement	Reverse Tournament Worst (RTW)
FSALPS Probability Calculation	Normal Frequency (NFC)

6.1.2 GP Language

The GP language used is not strongly typed since all terminal symbols are floating points. The selected language meets sufficiency requirements for a vision problem. Each non–terminal symbol is defined to meet closure, therefore operate error–free when supplied with floating–point arguments. The language is listed in Table 6.4. In each function, the first argument is represented as $arg0$ and second argument as $arg1$ $(arg0, arg1, arg2, ..., arg(n–1))$.

Terminals for the GP tree are obtained from pixels on the spectral bands. The Indian Pines hyperspectral data contains 200 spectral bands, which means using the pixel–based terminal data; we are, able to extract 200 spectral data for each pixel. An example of spectral data is shown in Figure 6.2 where each feature is obtained using $P_i(x, y)$ and converted to floating point between $0-1$ using Equation 6.1 where min is $minimum(P_0(x, y) - P_{200}(x, y))$ and max is $maximum(P_0(x, y) - P_{200}(x, y))$

$$T_i = P_i(x, y) - min/(max - min) \tag{6.1}$$

Table 6.4: Function set used for both representations.

Name	**Representation**	**Arity**	**Definition**
Maximum	max	2	$Maximum(arg0, arg1)$
Minimum	min	2	$Minimum(arg0, arg1)$
Multiplication	*	2	$arg0 * arg1$
Addition	+	2	$arg0 + arg1$
Subtraction	-	2	$arg0 - arg1$
Protective Division	%	2	$\begin{cases} arg1/arg0, & arg0 \neq 0 \\ 1, & arg0 = 0 \end{cases}$
Hyperspectral bands	b_1–b_{200}	0	pixel spectral from band k
Ephemeral Constant (ERC)	$-1 <= ERC <= 1$	0	Floating point constant

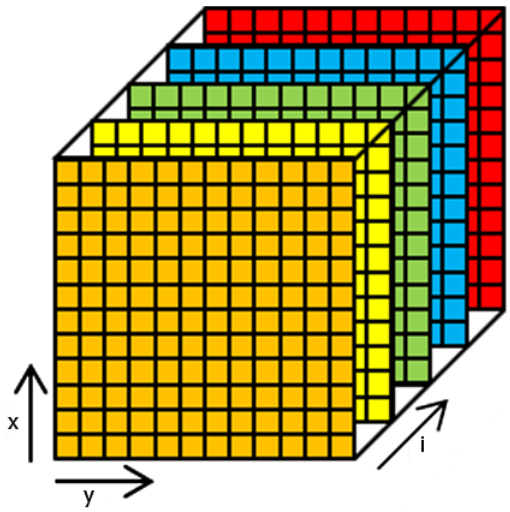


Figure 6.2: Hyperspectral bands

6.1.3 Fitness Function

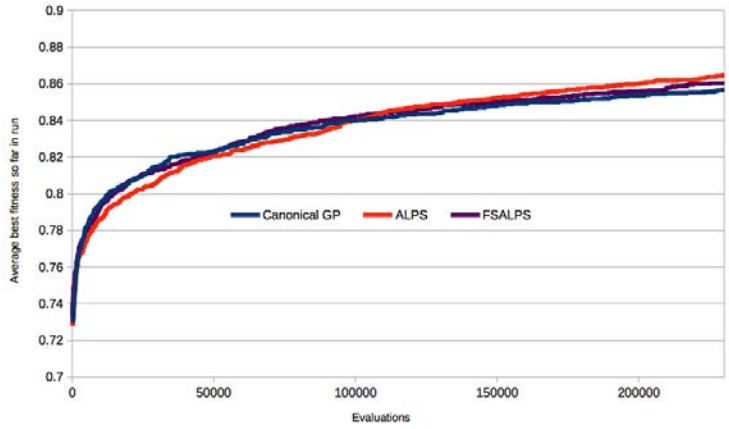
The fitness function is designed to award an individual that correctly classifies a pixel ($P_{x,y}$) where the target crop is resident. We do this by translating confusion matrix metrics into a fitness value (see Equation 6.2). Correctly identified crop features are *True Positives* (TP), correctly identified non–crop features are *True Negatives* (TN), incorrectly identified crop features are *False Positives* (FP) and incorrectly identified non–crop features are *false negatives* (FN). It is less effective using fitness functions that force output of GP to binary representation of features to be detected [60] therefore, in this work, standardized fitness (S_f) is the total number of wrongly classified examples ($FN + FP$).

$$Fitness = 1 - \frac{S_f}{\sum_{i=0}^{n} P_i(x, y)} \tag{6.2}$$

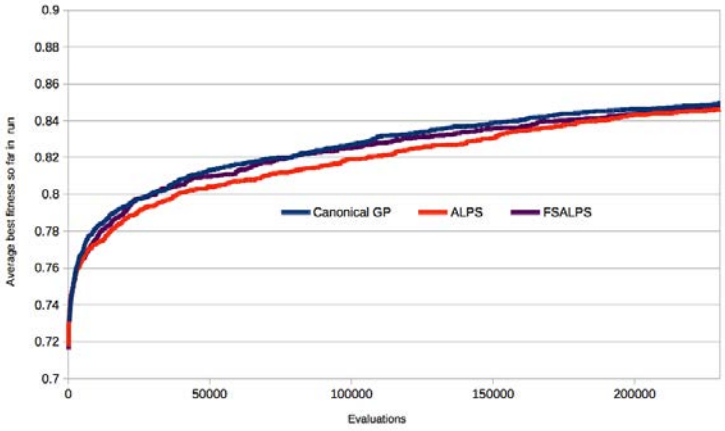
6.2 Results

We compare canonical, ALPS and FSALPS classification for the Indian pine hyperspectral data. Performance plots for training are shown in Figure 6.3. All three strategies did not show any signs of premature convergence on a mediocre solution. The best training individual for each strategy is applied to an unseen data, all performance metrics between canonical, ALPS and FSALPS are compared at 95% confidence interval using Tukey's HSD (honest significant difference) test [75]. In the result tables for the one–way ANOVA using Tukey's HSD , an arrow points in the

direction (row or column) of the strategy that is statistically dominant at the 95% confidence level.



(a) Corn–notill



(b) Soybean–mintil

Figure 6.3: Training performance plot

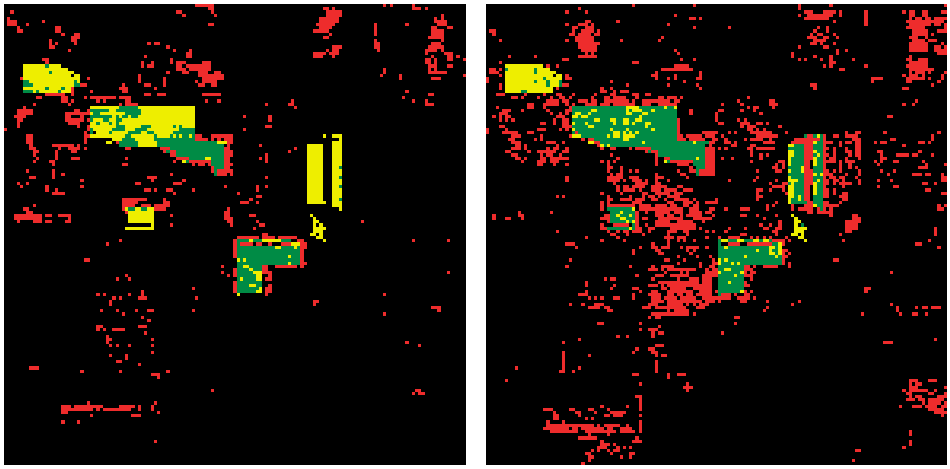
6.2.1 Classification Accuracy

The best–evolved classifier obtained from training is applied to the test data. Testing is done to show how the crop identifier classifiers unseen objects (pixels) on the entire field. The performance will also confirm if the evolved classifier is not over–fitted to the training data. The Corn–notill class contains 5.18% of the total samples available on each hyperspectral band while Soybean–mintil covers 18.30% per band of the hyperspectral cube. Performance plots (see Table 6.5 for interpretation of colors) showing how each algorithm performed on the hyperspectral image is shown in Figure 6.4, Figure 6.5 and Figure 6.6 for canonical, ALPS and FSALPS respectively. When considering Corn–notill, non–"Corn–notill"samples make up 94.82% of

the total hyperspectral data. GP could simply evolve a classifier that identifies only non–"Corn–notill"samples and still have an above average score. However our interest is in getting a crop identifier for the class under investigation. The raw percentage scores of classification accuracy are recorded in Table 6.6 for Corn–notill and Soybean–mintil. All three strategies performed very well on Corn–notill as opposed to Soybean–mintil. An ANOVA test using Tukey's HSD test at 95% confidence interval did not reveal any statistically significant difference between canonical, ALPS and FSALPS.

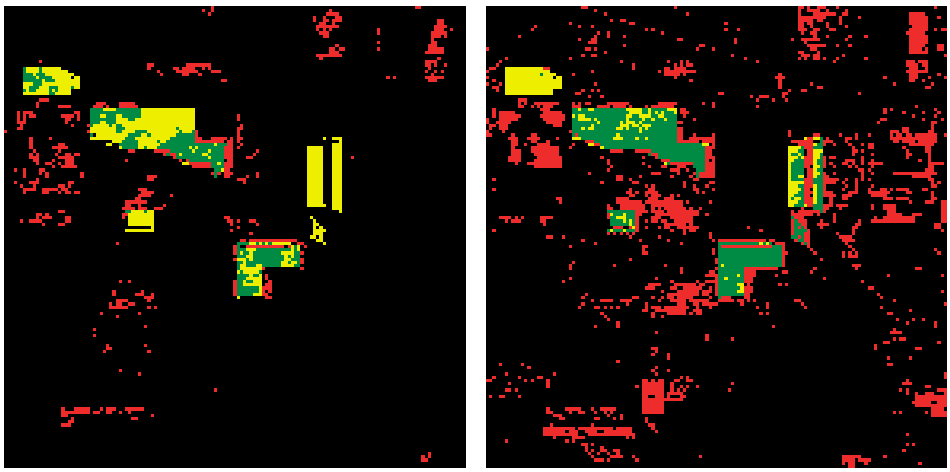
Table 6.5: Performance image legend.

Confusion Matrix	**Color Representation**
True Positive (TP)	Green
True Negative (TN)	Black
False Positive (FP)	Red
False Negative (FN)	Yellow



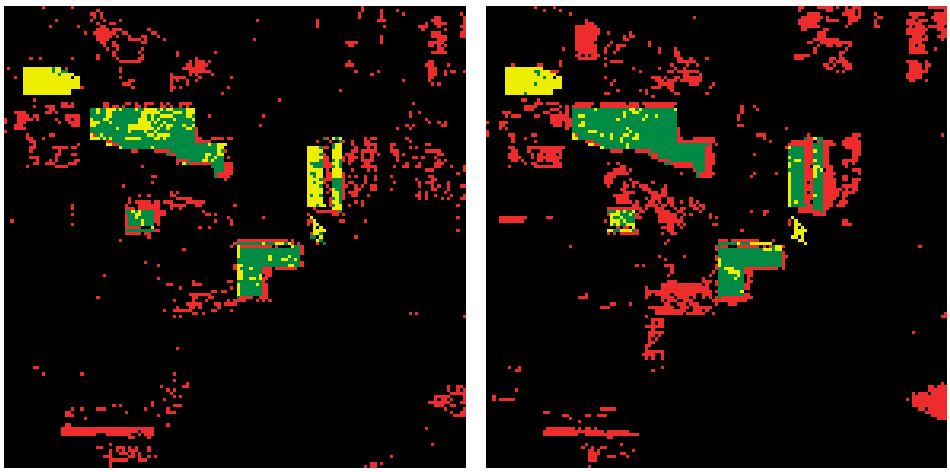
(a) Best canonical detection (b) Best canonical TP

Figure 6.4: Classification results for map area using canonical GP with (a) 92.51% classification accuracy with 40.09% detection of total TP and 95.39% detection of TN (b) highest detection of TP with 71.65% TP and 88.94% TN



(a) Best ALPS detection (b) Best ALPS TP

Figure 6.5: Classification results for map area using ALPS GP with (a) 93.69% classification accuracy with 37.06% detection of total TP and 96.78% detection of TN (b) highest detection of TP with 71.01% TP and 88.92% TN



(a) Best FSALPS detection (b) Best FSALPS TP

Figure 6.6: Classification results for map area using FSALPS GP with (a) 92.16% classification accuracy with 58.81% detection of total TP and 93.98% detection of TN (b) highest detection of TP with 74.50% TP and 90.97% TN

Given that the GP language used a single spectral property per pixel, without any spatial operators, the performance recorded for each algorithm seems good. The evolved classifiers in correctly identified difficult areas on the hyperspectral image such as similar crops, roadways with similarity to some crops, and boundaries of class regions under investigation. Some of these problems can be overcome by expanding the GP language to include multiple spectral properties for each pixel, as well as introducing spatial operators such as area filters.

FSALPS evolved more generalized solutions with highest detection of positive cases unlike canonical and ALPS that focused on detection of the much bigger area of negative examples on the Pines data. This result is consistent with both classes tested on the spectra data. A detailed discussion on classification accuracy of a second class on the Indian Pines hyperspectral dataset (soybean–mintil) is done in Appendix B.1.1.

	Canonical		ALPS		FSALPS	
	Average	Best	Average	Best	Average	Best
Corn–notill	87.07	92.52	88.02	93.69	86.64	92.16
Soybean–mintil	65.03	78.78	64.12	77.36	63.96	74.30

Table 6.6: Classification accuracy for best evolved crop identifiers on the Indian Pines hyperspectral data

6.2.2 Feature reduction

The hyperspectral dataset contains 200 features per spectral pixel. We seek to use a GP based evolutionary algorithm that significantly reduces the number of features required to accurately represent the problem. After applying all three algorithms to the problem set, Table 6.7 highlights key feature reduction data obtained. The data in Table 6.7 is obtained by analyzing the best solution tree for each strategy at the end of a run. By extracting information such as minimum, maximum and average of features contained within the best solution–tree, the summary table immediately shows that overall feature reduction is happening. This type of analysis is not biased towards the number of times a feature might have been used in bloat expressions but rather measures the availability of unique features in an individual.

Table 6.7: Minimum (Min), maximum(Max) , average (Avg) and Standard Deviation (StD) number of unique objectives found in the solution set for each dataset. Total objectives considered for each dataset including ERC is: Corn–notill 201 and Soybean–mintil 201

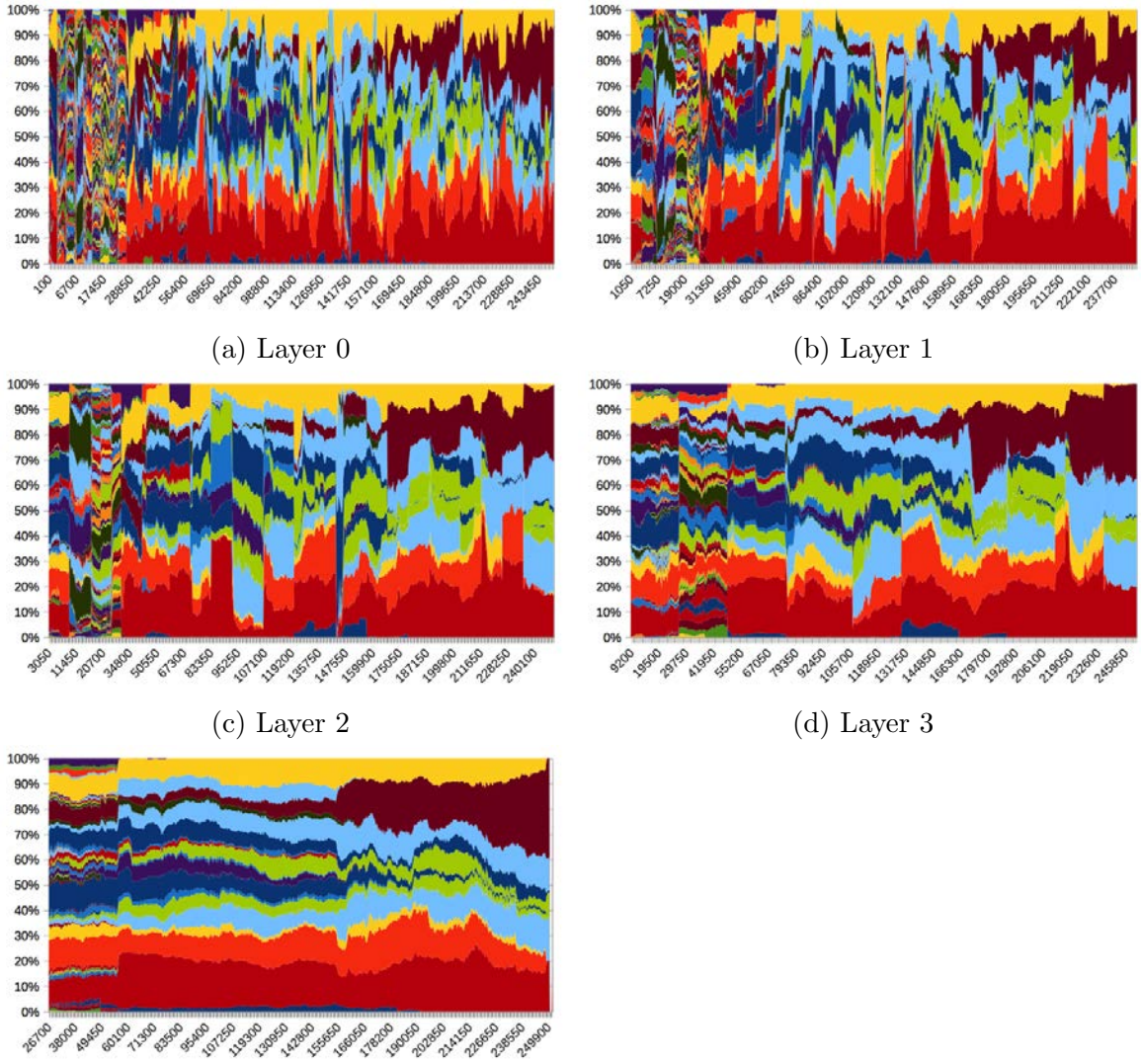
		Canonical	ALPS	FSALPS
Corn–notill	Min	18	31	7
	Max	52	84	35
	Avg	31.65	54.25	20.05
	StD	9.66	14.61	8.73
Soybean–mintil	Min	17	29	7
	Max	45	76	29
	Avg	29.7	55.7	20.95
	StD	9.02	10.42	6.64

Table 6.8: Comparing mean difference between feature selection performed for each strategy at the 0.05 significance level for Corn–notill($\otimes$) and Soybean–mintil($\oplus$)

	ALPS	FSALPS
Canonical	$\leftarrow^{\otimes\oplus}$	$\uparrow^{\otimes\oplus}$
ALPS	—	$\uparrow^{\otimes\oplus}$
FSALPS		—

We used an ANOVA test at 95% significance interval to compare feature reduction between all 20 runs, for all algorithms using Tukey's HSD one–way ANOVA test (see results in Table 6.8). FSALPS significantly outperformed canonical and ALPS for both crop types on feature reduction. On the other hand canonical GP outperformed ALPS in feature reduction. ALPS had the least feature reduction due to the regular introduction of new individuals into the bottom layer (constructed using random selection of terminal symbols). Although the random introduction of new features is responsible for ALPS ability to overcome premature convergence, it also re–introduces lots of features into the ALPS population after they had been previously removed. FSALPS addresses this shortfall in ALPS using its unique ability to perform tree construction using evolved feature probabilities. This means FSALPS is able to overcome the problem of premature convergence, perform feature ranking and construct

classifiers with high classification accuracy. The consistency in feature reduction by FSALPS is seen in the smaller values of standard deviation for all 20 runs, which proves that FSALPS becomes more stable with respect to feature selection.



(a) Layer 0

(b) Layer 1

(c) Layer 2

(d) Layer 3

(e) Layer 4

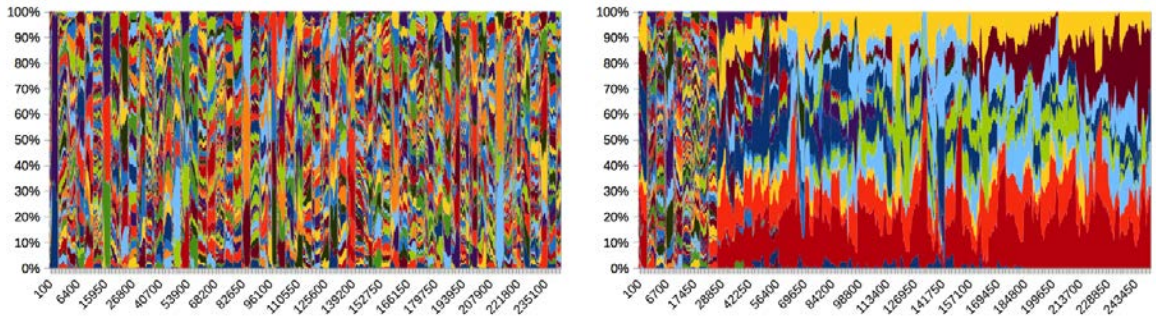
Figure 6.7: Percentage contribution of each feature per layer in the FSALPS–run with minimum number of selected features. Layer 4 ended with 7 features testing classification accuracy of 80.48%

Figure 6.7[3] overall contribution of features (each color band represents a feature) within an FSALPS layer. Evolution begins in layer 0 (Figure 6.7a) using equal probability distribution for all 201 features. Feature selection dynamics change when the first set of individuals is shipped to the last layer (Figure 6.7e) around evaluation

[3]additional plots for both classes are provided in Appendix B.2

26500. Once layer 4 becomes active, a feature count is initiated immediately before the next layer 0 restart. The feature count is converted to probabilities and used for feature selection during tree initialization or sub–tree construction during mutation operations. All other layers take shape from the evolved probability distribution. FSALPS ended with 7 out of the initial 201 features, thereby reducing the hypothesis space from 2^{201} to 2^{7}. Appendix B.2 discusses feature selection in best solution trees of canonical, ALPS and FSALPS.

Figure 6.8 compares layer 0 feature selection in ALPS and FSALPS. Selection of features in layer 0 of ALPS is always done randomly (see Appendix B for a discussion of ALPS feature selection). This illustrates why ALPS has the least feature selection among the three strategies. Canonical GP randomly selects features during tree initialization and only adds new features when performing mutation in higher generations. Introduction of new features in canonical GP through mutation is not guaranteed and depends on mutation probability (10% in this experiment). After initialization in canonical GP, new features are hardly introduced in higher canonical generations and could easily plunge the evolutionary system into an early convergence.



(a) ALPS Layer 0

(b) FSALPS Layer 0

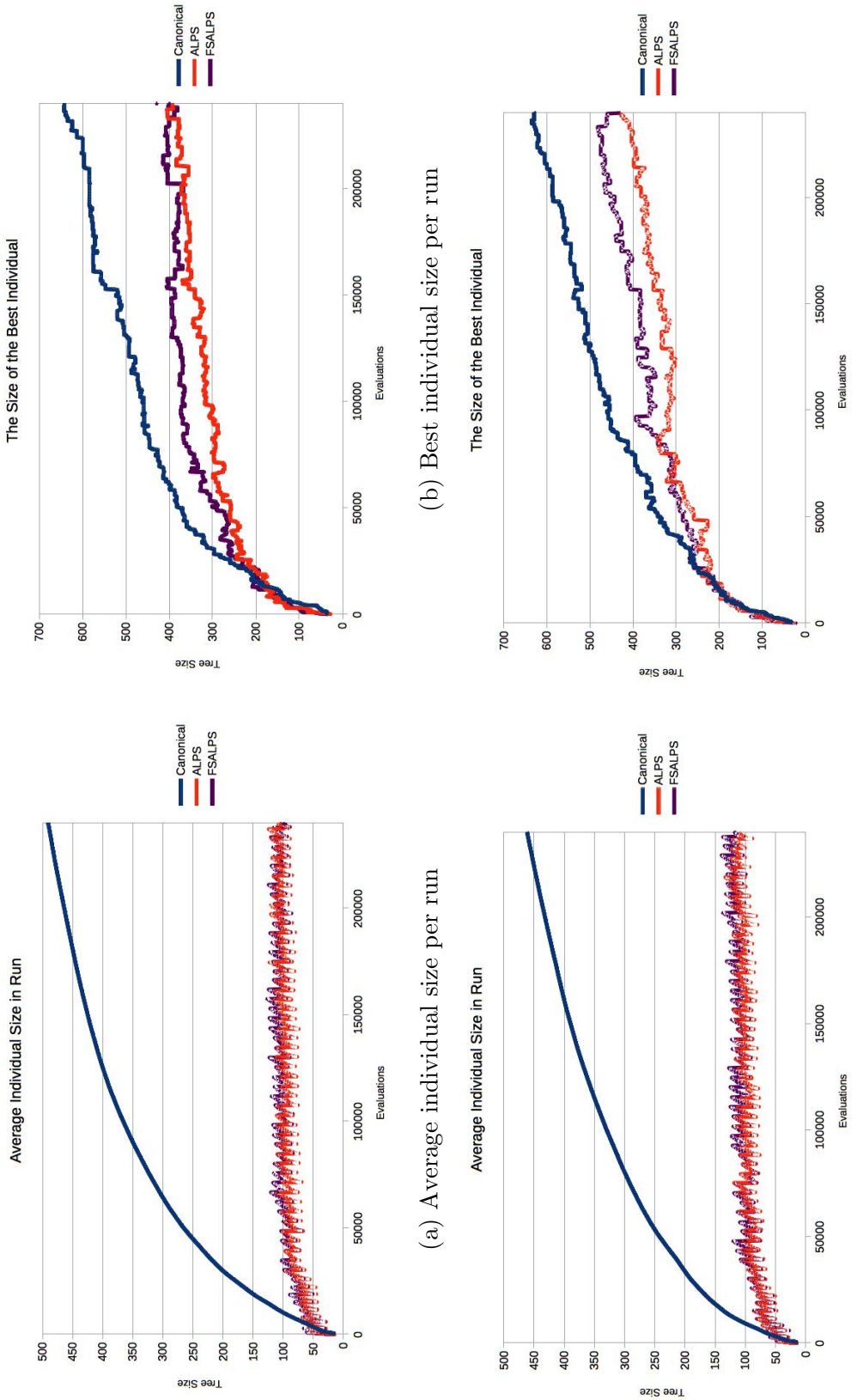
Figure 6.8: Percentage contribution of features in Layer 0 of (a) FSALPS and (b) ALPS. FSALPS is more stable than ALPS

6.2.3 Tree Size and Memory Usage

The size of a tree expression is measured by counting the number of nodes contained in a tree. The tree–size gives an idea of the memory requirements needed to run an algorithm. ALPS and FSALPS often produce very portable solutions as compared to canonical GP. The ALPS variants regularly (every age–gap generations) introduce new individuals into the bottom layer. Cross breeding between adjacent layers is allowed. Hybrid offspring (produced by breeding parent individuals from two adja-

cent layers) often transfer the small sized properties of bottom layer individuals to the higher layers. Figure 6.9 shows the rate of growth of solution trees within each strategy. Canonical has a higher rate of growth. In canonical GP, breeding is allowed between the entire population. The population is initialized once and fitness–based evolutionary selection pressure forces breeding between highly fit but often–large individuals.

In Table 6.9, we compared the difference in tree sizes using Tukey's HSD ANOVA test. The results confirm that the tree sizes of ALPS and FSALPS are significantly smaller than canonical GP. The results are consistent with both crop classes investigated in this work. See Appendix B.1.2 for further discussion on memory utilization by canonical, ALPS and FSALPS. Sample tree solutions are shown in Appendix B.3 as LISP S–Expressions.



(a) Average individual size per run

(b) Best individual size per run

(c) Average individual size per run

(d) Best individual size per run

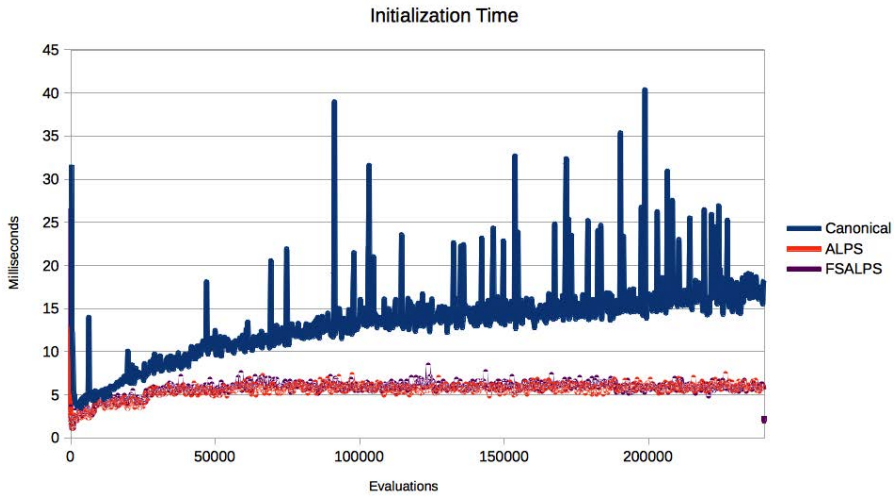
Figure 6.9: Tree growth using Corn–notill (a and b) and Soybean–mintil (c and d)

6.2.4 Time analysis

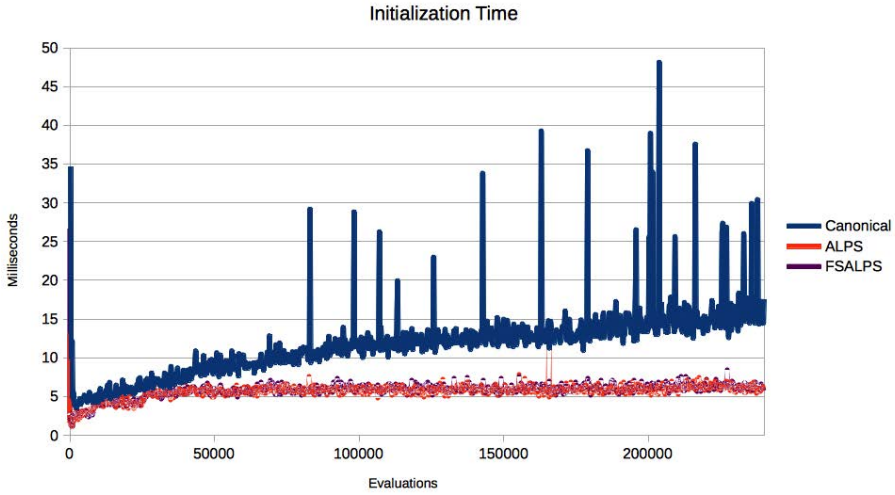
Two time components were analyzed —initialization (Figure 6.10) and execution time. Initialization time is the total time taken to breed the initial population. For subsequent generations, initialization time is how long the previous generation took to breed a new population. Evaluation time is the total time taken to evaluate a complete GP population. Both time factors are directly proportional to solution sizes. Table 6.9 reveals a significant difference between the three strategies with ALPS and FSALPS leading canonical at the 95% confidence interval. The time factor is consistent with the smaller tree sizes produced in ALPS and FSALPS.

Table 6.9: Comparing mean difference between best solution tree-size for each strategy at the 0.05 significance level for Corn–notill($\otimes$) and Soybean–mintil($\oplus$)

	ALPS	FSALPS
Canonical	$\uparrow^{\otimes\oplus}$	$\uparrow^{\otimes\oplus}$
ALPS	—	—$^{\otimes\oplus}$
FSALPS		—



(a) Corn–notill



(b) Soybean–mintil

Figure 6.10: Initialization time using 250,000 evaluations

6.3 Discussion

We used a number of criteria to measure the performance of the evolved classifiers towards identification of spectral features. A training individual with the highest classification accuracy after an evolutionary run is used to identify spectral features on untrained areas of the hyperspectral image. The GP language used was very basic yet classification accuracy for all three strategies was good. Using Tukey's HSD ANOVA test, we did not find a significant difference in classification accuracy at the 95% confidence interval. It was a surprise that feature reduction did NOT

result in significantly improved feature classification accuracy, contrary to common expectations. Future work might increase the language choice to include some spatial primitives. Also, expanding the language to include other spectral properties as done in [65] will improve identification of edges and increase the classification accuracy.

The classification function is able to discriminate a crop in the context of other crops or features available on the hyperspectral image. The training data was obtained by handpicking few samples distributed across the entire data. To further test the reliability of our constructed classifier, future work must consider testing the classifier on other similar geographical regions. This process will show the generality of the evolved classifier and prove that the obtained classifier is not fitted to the specified geographical region in this work. Another area that can be improved in future work is with regards to how data is obtained for training. Statistical sampling should be used to select samples that are adequate to represent various regions on the hyperspectral image.

FSALPS and ALPS are more efficient strategies than canonical GP in terms of time and space requirements. FSALPS and ALPS consistently produced smaller solution trees with smaller evaluation time when compared with canonical GP. The time and memory required to run either algorithms were significantly better as well.

FSALPS emerged as a dominant feature selection technique and recorded a more stable feature selection that is consistent with all runs. It has the unique ability to improve diversity while translating feature ranks into selection of features during tree initialization and sub–tree construction. Thus, FSALPS performs feature selection while avoiding premature convergence through random introduction of new individuals. The feature count strategy used in the FSALPS algorithm does not actively avoid bloat expressions and could reduce relevance of feature ranks generated from feature–probabilities. Future work in FSALPS can integrate a bloat scanning operation that avoids bloat expressions in a tree. Even though the language used in this work keeps bloat expressions at a minimal, an alternative improvement, (but more difficult strategy) is to come up with a GP language that prevents bloat.

It is difficult to say an algorithm is much better than the other [13]. However, given the above experiments, FSALPS emerged as superior when considering all benchmarks used in measuring performance of all three algorithms.

Chapter 7

Comparisons to related work

In this work, we examined the performance of canonical GP, ALPS and FSALPS on varied classification datasets. In this chapter, we include brief comparison of our work with related literature. The discrepancy in the comparisons we make with published data from related literature is informal and cannot be entirely relied upon due to varied parameter settings and experimental conditions.

7.1 Classification Experiments

In a related work, [26] used a variation of ALPS called spatial co–evolution ALPS (SCALPS). Although Harper[26] did not discuss how SCALPS performs on feature selection, SCALP was found to avoid bloat on a bivariate regression problem even though no express bloat reduction strategy was implemented. SCALP produced short, general solution trees on a classification dataset (human oral bioavailability pre–diction) as opposed to the larger trees produced by operator equalization and canonical GP, which happens to be consistent with our work. In this work, we found FSALPS and ALPS exhibiting both properties of smaller but highly generalizable trees on all classification data sets. In the work of [20], Pedro et al. identified computational cost and bloat as two main pitfalls associated with the use of GP-based classifiers. The ability of FSALPS and ALPS to produce smaller solution trees translated into less memory and time requirements needed to complete an evolutionary run. FSALPS emerged as the preferred choice due to its dual ability to perform feature reduction while reducing effects of bloat.

In [9] a modified version of GA called CHCGA[19] algorithm was used for feature selection with an external classifier. We compare our feature selection results to the wrapper based implementation of CHCGA for feature selection in [19] using

Pima Indians diabetes and Ionosphere datasets. Table 7.1 shows two variations of experiments done with the CHCGA strategy using two different classifiers. In both instances, FSALPS outperformed both strategies with higher classification accuracy and a smaller feature subset. In the Ionosphere dataset for instance, there were many unreported runs of FSALPS dominating the CHCGA strategy.

Table 7.1: Comparing feature reduction (Features) and Maximum Classification Accuracy(Max CA) experimental results between FSALPS and CHCGA with Support Vector Machine(SVM) and Radial Basis Function(RBF) classifiers[9]. Pima (Total = 9), Ionosphere (Total = 35)

Dataset		CHCGA + SVM	CHCGA + RBF	FSALPS	
Pima	*Features*	7	7	**5**	
	Max CA	80.47	76.82	**88.24**	
Ionosphere	*Features*	16	16	**9**	**11**
	Max CA	94.27	94.29	**94.44**	**100.0**

In a similar experiment, Smith et al.[71] used GP for feature creation with a GA feature selector. The GA and GP (GAP) based feature selection and classification relied on GP, C4.5, k-nearest neighbor, Naive Bayes classifiers for the Pima Indians Diabetes, Sonar, Ionosphere and Sonar dataset (all of which were used in this work). Best results obtained from [71] and other reported classification performance values captured in [71] are compared with FSALPS in Table 7.2. Table 7.2 and 7.3 compares reported feature selection and classification experiments in related literature. We show values of dominating strategies in bold with FSALPS dominating most of the reported results.

Table 7.2: Comparing performance of classifiers obtained in [71] with other reported results on Ionosphere(Total : 34), Pima Indians Diabetes (Pima, Total : 8), Sonar (Total : 60) and Wisconsin Breast Cancer –New (WBC New, Total : 30). For each of the datasets an ephemeral data set was added.

	Pima	WBC New	Ionosphere	Sonar
CHCGA+SVM[9]	80.47	–	94.27	–
CHCGA+RBF[9]	76.82	–	94.29	–
GAP (avg)[72]	75.72	96.14	89.90	85.57
Simple Meta[72]	76.04	95.09	89.82	86.64
GAP (best)[72]	79.62	98.86	96.17	96.42
C4.5[72]	67.94	–	–	69.69
HIDER[72]	74.10	–	–	56.93
XCS[72]	68.62	–	–	53.41
O.F.A[72]	69.80	–	–	79.96
LVSM[72]	78.12	–	–	–
Krawiec[72]	76.41	–	–	–
GAP (J48)[71]	73.64	95.71	90.69	75.89
GAP (IBK)[71]	68.96	94.82	91.38	83.72
GAP (NB)[71]	75.77	96.75	90.60	77.64
C4.5(J48)[71]	73.32	93.88	89.82	73.86
IBK[71]	69.90	95.44	86.95	86.65
N.B.[71]	75.13	93.26	82.37	67.16
FSALPS(avg)	74.03	93.82	90.65	74.36
FSALPS(best)	**94.12**	**100.0**	**100.0**	**100.0**

Table 7.3: Comparing feature selection of some published strategies and FSALPS using Ionosphere(Iono., Total : 34), Pima Indians Diabetes (Pima, Total : 8), Sonar (Total : 60) and Wisconsin Breast Cancer - New (WBC New, Total : 30). For each of the datasets an ephemeral data set was added.

Research		Pima	WBC New	Ionosphere	Sonar
CHCGA + SVM [9]	Min	7	–	16	–
CHCGA + RBF [9]	Min	7	–	16	–
[72]	Min	–	–	–	–
	Avg	**6.4**	15.7	19.7	38.0
FSALPS	Min	**3**	**5**	**7**	**7**
	Avg	**6.4**	**9.3**	**12.75**	**16.85**

7.2 Hyperspectral

We handpicked our training examples just as was done in [65, 62]. Ross et al.[65] used canonical GP to evolve mineral identifiers for the Cuprite area. FSALPS achieved remarkable classification accuracy using a very simple GP language without having to use thresholds that maximize presence of target class labels as done in [65]. In our results, FSALPS shows smaller trees, but less sophisticated visual classification language.

Le et al.[47] used a hybrid GA and SVM system to perform band reduction on two benchmark hyperspectral data sets — Washington DC Mall data set and the Indian Pine data set. Their work used 202 out of the original 220 bands by eliminated 18 affected bands. We proceed to compare our work with the related work reported by Le et. al.[47]. The comparison done here is not entirely accurate due to varying experimental conditions and different parameter settings. Some of the notable disparities arise from the use of 10–fold while we used 20–fold, multi–class classification as against single-class classification, etc. We assume the results presented in their report are reflective of the average performance per class on the Pima Indians hyperspectral image. We include in Table 7.4 a comparison of three search algorithms proposed by Serpico et al.[68] that was tested on the Indian Pines hyperspectral dataset and reported by [47]. Le et al.[47] differ from Serpico et al.[68] mainly in the classifiers employed in their work –the former used SVM and the latter used a MAP classifier.

Table 7.4: Comparing feature(spectral bands) selection of FSALPS corn–notil class with related work on Indian Pines hyperspectral data.

Algorithms	Number of bands	Min accuracy	Max accuracy	Avg Accuracy
All bands(SVM)[47]	202	83.29	83.29	83.29
SFBE (SVM)[47]	26	86.59	86.59	86.59
FCBE (SVM)[47]	26	86.48	86.48	86.48
SABE (SVM)[47]	28	86.64	86.64	**86.64**
CGGS + BB[47]	12	83.32	84.66	83.95
SFBE[68]	26	81.30	81.30	81.30
FCBE[68]	26	81.46	81.46	81.46
SABE[68]	28	81.57	81.57	81.57
FSALPS[1]	20.05	72.16	**92.16**	**86.64**

The average number of bands used in FSALPS is consistent with reported values in related research[5, 24, 68], that shows that a minimum of 20 bands are needed to accurately represent the Indian Pines hyperspectral dataset. FSALPS has shown great success as a feature reduction and classification algorithm. We have included key future work suggestions that will further improve the performance indicators.

Chapter 8

Conclusions and Future Work

In this work, we have exploited the dynamic representational formalism of GP to perform feature selection and classification. In this chapter, we will summarize the various findings obtained from the different experiments and look at possible future directions to this research.

8.1 Conclusion

GP is an adaptive machine learning technique that does not require expert domain knowledge to represent a problem. We showed this by using GP to represent various classification problems including Pima Indians diabetes[48], breast cancer diagnostic[48], ionosphere[48], sonar[48] and the Indian Pines hyperspectral image[64]. We compared the performance of canonical, ALPS and FSALPS using some key performance indicators; the results were consistent for all datasets.

The main goal of this research is to compare how the various strategies performed on feature selection and classification problems. We were looking for an efficient strategy that selected relevant subset from a given feature vector – this subset should be more representative of the problem and produce a classification accuracy that is at least as good as the original feature vector. FSALPS emerged dominant over canonical and ALPS as a preferred feature selection strategy in terms of a significant reduction in the size of the feature subset used to represent the original feature vectors. FSALPS is thus the best alternative when considering a preprocessing technique for a data mining and knowledge discovery task. On the other hand, ALPS emerged as the least effective in terms of feature reduction. This is explained by the random introduction of new features without necessarily considering the relevance of those features. The bottom layer is very noisy and migrates its individuals through higher

layers. The one–time initialization that happens in canonical GP allows features to evolve through higher generations. Meanwhile, restricting the entire population to features selected during initialization increases the probability of converging the entire population on a mediocre fitness, especially if some relevant features are missing in the population. Mutation operations in canonical GP seek to address this problem through the random selection of features to construct sub–trees. Given the low rate of mutation (mutation rate is usually kept low to prevent random walk), feature selection is ineffective.

FSALPS overcomes these inherent limitations in canonical GP and ALPS by combining the strengths of both algorithms. More importantly, FSALPS is powered by a novel feature probability calculation that does a directed feature search without completely eliminating a feature. Rather, FSALPS directs the search process by using the evolved feature frequencies in the selection of terminals for the construction of GP tree expressions. It is very important that the system does not completely eliminate any feature to forestall possible premature convergence. Also, feature selection in FSALPS is very stable in all experiments when compared to canonical and ALPS. We are able to conclude that the individual features in the best–evolved solutions of FSALPS are relevant because the FSALPS strategy produced highly competitive classification accuracy to canonical and ALPS.

Surprisingly, FSALPS solutions did not score significantly higher classification accuracy than canonical and ALPS GP contrary to common expectations that simpler search space means more accurate classifiers. Canonical GP and ALPS are able to compensate and sacrifice feature selection for classification accuracy.

We also analyzed the resource requirements for running canonical GP, ALPS and FSALPS GP. The time used to complete an ALPS and FSALPS run was significantly lower than that of canonical GP. ALPS and FSALPS regularly introduced new individuals into the bottom layer. These small individuals quickly migrate into higher layers and very often end up replacing some of their larger ancestors. Consequently, small but effective individuals from different fitness basins reside through the entire layered–population. Due to the restricted inter–layer breeding imposed on ALPS and FSALPS, genetic operations (such as crossover) usually breed hybrid offspring individuals that are a result of separate fitness basins. The tree size factor remains fairly stable because of slight differences in observed tree–size in adjacent layers of the ALPS variants. The stabilization in growth of tree–size observed in higher generations of ALPS and FSALPS translated into the stable and reduced time required to complete an ALPS and FSALPS run. The minimal time and space complexity

requirement observed for ALPS and FSALPS were consistent in all datasets, thus emphasizing the efficient nature of these strategies.

8.2 Future Work

We have shown FSALPS as an effective feature reduction strategy, however we have noted some shortfalls that if addressed, could improve the feature selection and classification performance of FSALPS.

8.2.1 Detection of Bloat

The feature ranking system used in FSALPS relies on feature–frequency counts performed on the population. The feature–frequencies will be more representative relevant of features if bloat expressions are actively avoided during feature counts. We have implemented some strategies in our tree based representation to reduce the occurrence of bloat such as the refinement of GP language and parsimony pressure. Although it will be difficult to completely eliminate bloat expressions, new strategies could further reduce or minimize their influence on feature counts.

1. **Size Fair Crossover:**

   Two parents are selected for cross breeding using fitness–based tournament selection[42]. A crossover point is determined in the first parent and the size of the sub–tree at the crossover point is calculated. The crossover point from the second parent is selected such that the sub–tree is of the same size as the sub–tree selected from the first parent. In our work, we empirically determined a crossover rate of $90 + \%$, which means breeding often takes place through crossover. Applying this bloat reduction mechanism involves choosing the right size parent and repeated traversal of sub–trees of the second parent to select the optimal crossover point. The computational overhead is expensive and will significantly increase breeding time for FSALPS. Since one key aspect of this work was to reduce computational time, the size fair crossover was not used, and so we relied on other less expensive strategies.

2. **Program simplifier:** This approach involves the use of a program that is called after breeding to scan and eliminate bloat expressions from all individuals. The scan and elimination process will shoot up computational requirements and render the algorithm more expensive to execute.

8.2.2 Re–run Using Reduced Feature Set

Future work could use the reduced feature as a base feature set to run an experiment on the same problem. With fewer features, the search space is reduced and solutions may be easier to find.

8.2.3 Incorporating Feature Extraction into FSALPS

Feature mining includes feature selection and feature extraction. Feature extraction is a well–known pre–processing technique that is used in data mining tasks[9, 20]. There are situations where the original feature vectors are not representative of the given problem and will require the discovery of interesting hidden relationships between the features. GP has the unique ability to construct such composite features by evaluating the tree expressions to produce a linear or non–linear combination of the original feature vector. Thus it is possible to exploit GPs representational power to construct a hybrid FSALPS system that performs feature selection and feature extraction as done in [49]. We believe the hybrid system will facilitate the discovery of new relevant features leading to improved classification accuracy and a more refined feature subset.

8.2.4 GP Language for Hyperspectral Image

In our work on the Indians Pines hyperspectral data, we used a very basic terminal language that represented raw pixel values obtained from the various spectral bands. Improved results may arise when spatial operators and other spectral properties[65] are added to the GP language. These additions could easily facilitate detection of more difficult regions (e.g. boundaries and edges) on the hyperspectral image.

8.2.5 Multi–Classification

Most of the problems investigated in this work involved binary classifications. The hyperspectral data considered was a multi–classification problem that was handled as binary classification by evolving crop–identifiers for each crop of interest. Given the search power and feature selection ability of FSALPS, it will be interesting to directly tackle multi–class problems using FSALPS. A successful representation of a multi–class problem in a GP domain means we are able to evolve a GP expression that simultaneously discriminates multiple classes on the hyperspectral data. This will lead to much simplified solution for multi–class problems.

8.2.6 Fitness Function

We used classification accuracy to measure fitness of GP individuals. There are a number of criteria that could be used to evaluate fitness of a feature selection problem, such as sensitivity, specificity, information gain, maximum relevance, feature correlation and mutual information[9]. Some of these performance indicators measure performance of an entire feature subset, or individual feature relevance based on how unique features discriminate class labels, or rank a feature based how it compares to other features. It will be interesting to know how a multi–objective fitness evaluation combines a number of such criteria to score GP individuals without introducing any biases. By considering multiple fitness objectives, it should be possible to attain a feature set with more relevant features and a corresponding higher prediction of class labels.

8.2.7 Frequency Count for Non-terminal Symbols

In this work, we performed frequency count for only terminal symbols (features). Since a GP expression is composed of terminals and non–terminals, another dimension to this research is to investigate how ranking of non–terminal symbols will direct the search process. Such ranks are then used during construction of tree expressions. The implementation of this strategy will mean a holistic evolution of relevant nodes in the entire population. Another merit of such an implementation will be improvement in classifiers or extracted features (when performing feature extraction).

Bibliography

[1] D.P.M. Cobo D..R.P. Utrero Aguilar, D.P.K. and D.M.A.H. Nieves. Abundance extractions from aviris image using a self-organizing neural network. *In: Proceedings of the Ninth Annual JPL Airborne Earth Science Workshop.*, 5, 2000.

[2] Enrique Alba, José García-Nieto, Laetitia Jourdan, and El-Ghazali Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *IEEE Congress on Evolutionary Computation*, pages 284–290. IEEE, 2007.

[3] Khaled Badran and Peter Rockett. Multi-class pattern classification using single, multi-dimensional feature-space feature extraction evolved by multi-objective genetic programming and its application to network intrusion detection. *Genetic Programming and Evolvable Machines*, 13(1):33–63, 2012.

[4] Y. Bazi and F. Melgani. Toward an optimal svm classification system for hyperspectral remote sensing images. *Geoscience and Remote Sensing, IEEE Transactions on*, 44(11):3374–3385, Nov 2006.

[5] Y. Bazi and F. Melgani. Toward an optimal svm classification system for hyperspectral remote sensing images. *Geoscience and Remote Sensing, IEEE Transactions on*, 44(11):3374–3385, Nov 2006.

[6] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, December 1997.

[7] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[8] S. Cateni, V. Colla, and M. Vannucci. General purpose input variables extraction: A genetic algorithm based procedure give a gap. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 1278–1283, Nov 2009.

[9] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Comput. Electr. Eng.*, 40(1):16–28, January 2014.

[10] Park H. Lee J. Chi-Hyuck J., Lee S. *Use of partial least squares regression for variable selection and quality prediction*, volume computers & Industrial Engineering. Cie 2009, 6-9 july edition, 2009.

[11] C. Chion, J.-A. Landry, and L. Da Costa. A genetic-programming-based method for hyperspectral data information extraction: Agricultural applications. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(8):2446–2457, Aug 2008.

[12] Minshan Cui, S. Prasad, Wei Li, and L.M. Bruce. Locality preserving genetic algorithms for spatial-spectral hyperspectral image classification. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 6(3):1688–1697, June 2013.

[13] Joseph C. Culberson. On the futility of blind search: An algorithmic view of no free lunch. *Evol. Comput.*, 6(2):109–127, June 1998.

[14] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1:131–156, 1997.

[15] Liu H. Dash M. *Feature Selection for Classification Using Genetic Algorithms with a Novel Encoding*, volume 9th International Conference, CAIP 2001 Warsaw, Poland, September 57. Springer Berlin Heidelberg, 2001.

[16] Goldberg D.E. *Genetic Algorithms in Search Optimization and Machine Learning*. Addition Wesley, 1989.

[17] JeferssonAlex dos Santos, AndréTavares da Silva, Ricardo da Silva Torres, AlexandreXavier Falcão, LéoP. Magalhães, and RubensA.C. Lamparelli. Interactive classification of remote sensing images by using optimum-path forest and genetic programming. In Pedro Real, Daniel Diaz-Pernil, Helena Molina-Abril, Ainhoa Berciano, and Walter Kropatsch, editors, *Computer Analysis of Images and Patterns*, volume 6855 of *Lecture Notes in Computer Science*, pages 300–307. Springer Berlin Heidelberg, 2011.

[18] Anikó Ekárt and András Márkus. Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms. *Artif. Intell. Eng. Des. Anal. Manuf.*, 17(3):205–220, June 2003.

[19] Eshelman. The CHC Adaptive Search Algorithm : How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. *Foundations of Genetic Algorithms*, pages 265–283, 1991.

[20] Pedro G. Espejo, Sebastián Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *Trans. Sys. Man Cyber Part C*, 40(2):121–144, March 2010.

[21] Purdue Research Foundation. Multispec: A freeware multispectral image data analysis system, 2015. `https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html` [Online; Accessed 24-April-2015].

[22] Huanzhang Fu, Zhongzhe Xiao, Emmanuel Dellandréa, Weibei Dou, and Liming Chen. Image categorization using esfs: A new embedded feature selection method based on sfs. In Jacques Blanc-Talon, Wilfried Philips, Dan C. Popescu, and Paul Scheunders, editors, *ACIVS*, volume 5807 of *Lecture Notes in Computer Science*, pages 288–299. Springer, 2009.

[23] Piotr S. Gromski, Yun Xu, Elon Correa, David I. Ellis, Michael L. Turner, and Royston Goodacre. A comparative investigation of modern feature selection and classification approaches for the analysis of mass spectrometry data. *Analytica Chimica Acta*, 829(0):1 – 8, 2014.

[24] Baofeng Guo, R.I. Damper, Steve R. Gunn, and J.D.B. Nelson. A fast separability-based feature-selection method for high-dimensional remotely sensed image classification. *Pattern Recognition*, 41(5):1653 – 1662, 2008.

[25] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[26] Robin Harper. Spatial co-evolution: quicker, fitter and less bloated. In Terence Soule and Jason H. Moore, editors, *GECCO*, pages 759–766. ACM, 2012.

[27] Tibshirani R Hastie TJ, Buja A. Penalized discriminant analysis. *Ann. Statist.*, 23(1)(2):73–102, 04 1995.

[28] Mingyi He, Yifan Zhang, Yuzhen Xie, Na Liang, and Changyun Wen. Classification of multi-spectral/hyperspectral data using genetic programming and error-correcting output codes. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–6, May 2006.

[29] Gregory Hornby. Alps: the age-layered population structure for reducing the problem of premature convergence. In Mike Cattolico, editor, *GECCO*, pages 815–822. ACM, 2006.

[30] Gregory S. Hornby. Steady-state alps for real-valued problems. In Franz Rothlauf, editor, *GECCO - In Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 795–802. ACM, 2009.

[31] Jian Jun Hu and Erik D. Goodman. The hierarchical fair competition (hfc) model for parallel evolutionary algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation: CEC2002, (forthcoming), IEEE*, 2002.

[32] Jianjun Hu, Erik D. Goodman, Kisung Seo, and Min Pei. Adaptive hierarchical fair competition (ahfc) model for parallel evolutionary algorithms. In *In Proceedings of the Genetic and Evolutionary Computation Conference: 772-779, GECCO-2002*, 2002.

[33] Jianjun Hu, Erik D. Goodman, Kisung Seo, and Key Words. Continuous hierarchical fair competition model for sustainable innovation in genetic programming. In *In Genetic Programming Theory and Practice, Kluwer*, pages 81–98, 2003.

[34] Marcus Hutter. Fitness uniform selection to preserve genetic diversity. In *In Proc. 2002 Congress on Evolutionary Computation (CEC-2002*, pages 783–788. IEEE, 2002.

[35] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:153–158, 1997.

[36] Xiuping Jia, Bor-Chen Kuo, and M.M. Crawford. Feature mining for hyperspectral image classification. *Proceedings of the IEEE*, 101(3):676–697, March 2013.

[37] Thanyaluk Jirapech-Umpai and J. Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6:148, 2005.

[38] Koza J.R. *Genetic Programming.* MIT Press., 1992.

[39] George Kapetanios. Variable selection in regression models using nonstandard optimisation of information criteria. *Computational Statistics & Data Analysis*, 52(1):4–15, 2007.

[40] Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, December 2002.

[41] ThomasNavin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. Embedded methods. In Isabelle Guyon, Masoud Nikravesh, Steve Gunn, and LotfiA. Zadeh, editors, *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 137–165. Springer Berlin Heidelberg, 2006.

[42] W.B. Langdon. Size fair and homologous tree crossovers for tree genetic programming. *Genetic Programming and Evolvable Machines*, 1(1-2):95–119, 2000.

[43] WilliamB. Langdon. Large scale bioinformatics data mining with parallel genetic programming on graphics processing units. In Francisco Fernández de Vega and Erick Cantú-Paz, editors, *Parallel and Distributed Computational Intelligence*, volume 269 of *Studies in Computational Intelligence*, pages 113–141. Springer Berlin Heidelberg, 2010.

[44] Martin H. C. Law, Mário A. T. Figueiredo, and Anil K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE TRANS. PATTERN ANAL. MACH. INTELL*, 26(9):1154–1166, 2004.

[45] Michal Lemczyk and Malcolm Heywood. Pareto-coevolutionary genetic programming classifier. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 945–946, New York, NY, USA, 2006. ACM.

[46] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. pages 4–15. Springer Verlag, 1998.

[47] Shijin Li, Hao Wu, Dingsheng Wan, and Jiali Zhu. An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine. *Knowledge-Based Systems*, 24(1):40 – 48, 2011.

[48] M. Lichman. UCI machine learning repository, 2013. `http://archive.ics.uci.edu/ml` [Online; Accessed: 2 November 2014].

[49] Jung-Yi Lin, Hao-Ren Ke, Been-Chian Chien, and Wei-Pang Yang. Classifier design with feature selection and feature extraction using layered genetic programming. *Expert Syst. Appl.*, 34(2):1384–1393, February 2008.

[50] Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007.

[51] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):491–502, April 2005.

[52] Hall M.A. and Smith L.A. *Feature Selection Subset Selection: A Correlation Based Filter Approach*, volume Proc. International Conference on Neural Information Processing and Intelligence Information System,. In Springer (Ed), 1997.

[53] Ronen Meiri and Jacob Zahavi. Using simulated annealing to optimize the feature selection problem in marketing applications. *European Journal of Operational Research*, 171(3):842–858, 2006.

[54] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[55] Frederick Mosteller. A k-sample slippage test for an extreme population. *Ann. Math. Statist.*, 19(1):58–65, 03 1948.

[56] Olly Oechsle and Adrian F. Clark. Feature extraction and classification by genetic programming. In *Proceedings of the 6th International Conference on Computer Vision Systems*, ICVS'08, pages 131–140, Berlin, Heidelberg, 2008. Springer-Verlag.

[57] Oracle. Data mining concepts. `http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_decisiontree.htm#BGBDGCFI` [Online; Accessed: 18 June 2014].

[58] Fernando E. B. Otero, Monique M. S. Silva, Alex A. Freitas, and Julio C. Nievola. Genetic programming for attribute construction in data mining. In *Proceedings of the 6th European Conference on Genetic Programming*, EuroGP'03, pages 384–393, Berlin, Heidelberg, 2003. Springer-Verlag.

[59] Suneer Patel and Christopher D. Clack. Alps evaluation in financial portfolio optimisation. In *IEEE Congress on Evolutionary Computation*, pages 813–819. IEEE, 2007.

[60] Riccardo Poli. Genetic programming for image analysis. In *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 363–368, Cambridge, MA, USA, 1996. MIT Press.

[61] Warren Buckler Powell. *Approximate Dynamic Programming - Solving the Curses of Dimensionality.* Wiley, 2007.

[62] Patrick J. Rauss, Jason M. Daida, and Shahbaz Chaudhary. Classification of spectral imagery using genetic programming. In Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 726–733, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[63] Michael L. Raymer, William F. Punch III, Erik D. Goodman, Leslie A. Kuhn, and Anil K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evolutionary Computation*, 4(2):164–171, 2000.

[64] Manuel Graña Romay. Hyperspectral remote sensing scenes, 2015. `http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes` [Online; Accessed: 20 April 2015].

[65] Brian J. Ross, Anthony G. Gualtieri, Frank Fueten, and Paul Budkewitsch. Hyperspectral image analysis using genetic programming. *Appl. Soft Comput.*, 5(2):147–156, 2005.

[66] G. Balan S. Paus Z. Skolicki E. Popovici J. Harrison J. Bassett R. Hubley S. Luke, L. Panait and A. Chhircop. Ecj: A java-based evolutionary computation research system, version 22, 06 2000-2015. `http://www.cs.gmu.edu/~eclab/projects/ecj/` [Online; Accessed: 2 April 2014].

[67] Michael Schmidt and Hod Lipson. Age-fitness pareto optimization. In Rick Riolo, Trent McConaghy, and Ekaterina Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer New York, 2011.

[68] S.B. Serpico and G. Moser. Extraction of spectral channels from hyperspectral images for classification purposes. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(2):484–495, Feb 2007.

[69] Karel Slaný. Comparison of cgp and age-layered cgp performance in image operator evolution. In *Proceedings of the 12th European Conference on Genetic Programming*, EuroGP '09, pages 351–361, Berlin, Heidelberg, 2009. Springer-Verlag.

[70] Lindsay I Smith. A tutorial on principal components analysis. Technical report, Cornell University, USA, February 26 2002.

[71] Matthew G. Smith and Larry Bull. Using genetic programming for feature creation with a genetic algorithm feature selector. In Xin Yao, Edmund K. Burke, José Antonio Lozano, Jim Smith, Juan J. Merelo Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *PPSN*, volume 3242 of *Lecture Notes in Computer Science*, pages 1163–1171. Springer, 2004.

[72] MatthewG. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[73] Randall B. Smith. Introduction to hyperspectral imaging, 2012. `http://www.microimages.com/documentation/Tutorials/hyprspec.pdf` [Online; Accessed: 20 April 2015].

[74] Dong Song, MalcolmI. Heywood, and A.Nur Zincir-Heywood. A linear genetic programming approach to intrusion detection. In Erick Cantú-Paz, JamesA. Foster, Kalyanmoy Deb, LawrenceDavid Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, MitchA. Potter, AlanC. Schultz, KathrynA. Dowsland, Natasha Jonoska, and Julian Miller, editors, *Genetic and Evolutionary Computation — GECCO 2003*, volume 2724 of *Lecture Notes in Computer Science*, pages 2325–2336. Springer Berlin Heidelberg, 2003.

[75] John W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114, June 1949.

[76] Wikipedia. Cross-validation (statistics), 2012. `https://en.wikipedia.org/wiki/Cross-validation_(statistics)` [Online; Accessed: 20 March 2015].

[77] Wikipedia. Confusion matrix, 2015. `https://en.wikipedia.org/wiki/Confusion_matrix` [Online; Accessed: 24 January 2015].

[78] Joshua Wu. k-fold cross validation. `http://www.csie.ntu.edu.tw/~b92109/course/Machine%20Learning/Cross-Validation.pdf` [Online; Accessed: 17 April 2015].

[79] Jihao Yin, Yifei Wang, and Jiankun Hu. A new dimensionality reduction algorithm for hyperspectral image using evolutionary strategy. *Industrial Informatics, IEEE Transactions on*, 8(4):935–943, Nov 2012.

[80] Mandi Zhou, Jiong Shu, and Zhigang Chen. Classification of hyperspectral remote sensing image based on genetic algorithm and svm, 2010.

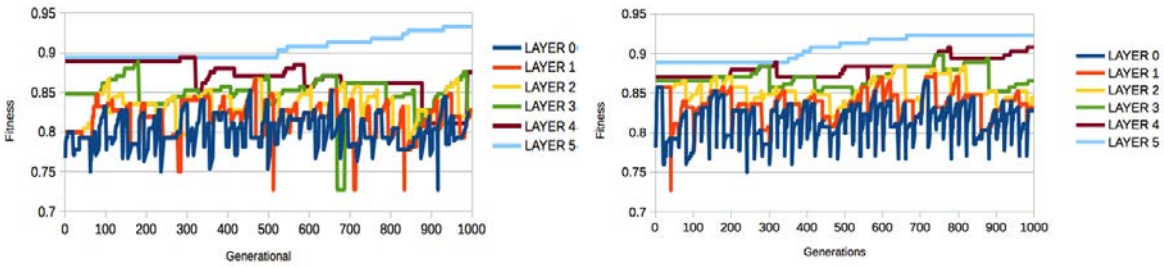
[81] Li Zhuo, Jing Zheng, Xia Li, Fang Wang, Bin Ai, and Junping Qian. A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine, 2008.

Appendix A

Further Experimental Analysis

A.1 ALPS Setup

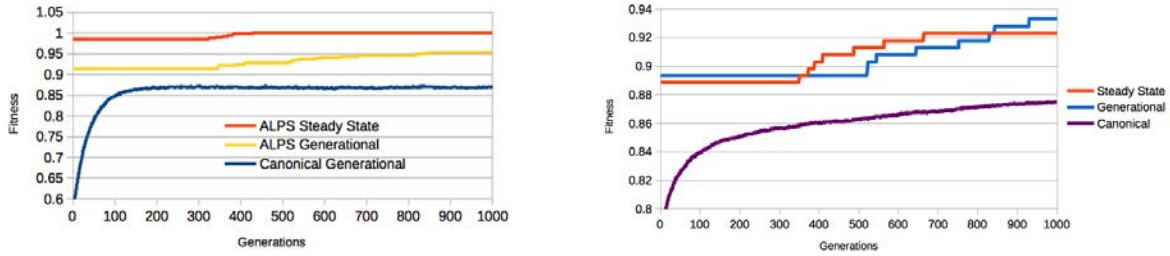
In Figure A.1, six ALPS layers were run using the Pima Indians Diabetes dataset. Evolution starts from the bottom (layer 0) and progresses until individuals are old enough to migrate to layer 1. This usually happens for every "age-gap"generations for generational replacement or "age–gap * layer–0 population–size"evaluations for steady state replacement. Highly fit and older individuals automatically bubble into higher layers and can be seen in Fig A.2. Only the best individual in the last layer (Layer 5 Figure A.1) is guaranteed to stay until a stronger individual from the bottom layer replaces it.



(a) Generational Replacement

(b) Steady State Replacment

Figure A.1: ALPSGP on Pima Indians Diabetes dataset using two replacement strategies



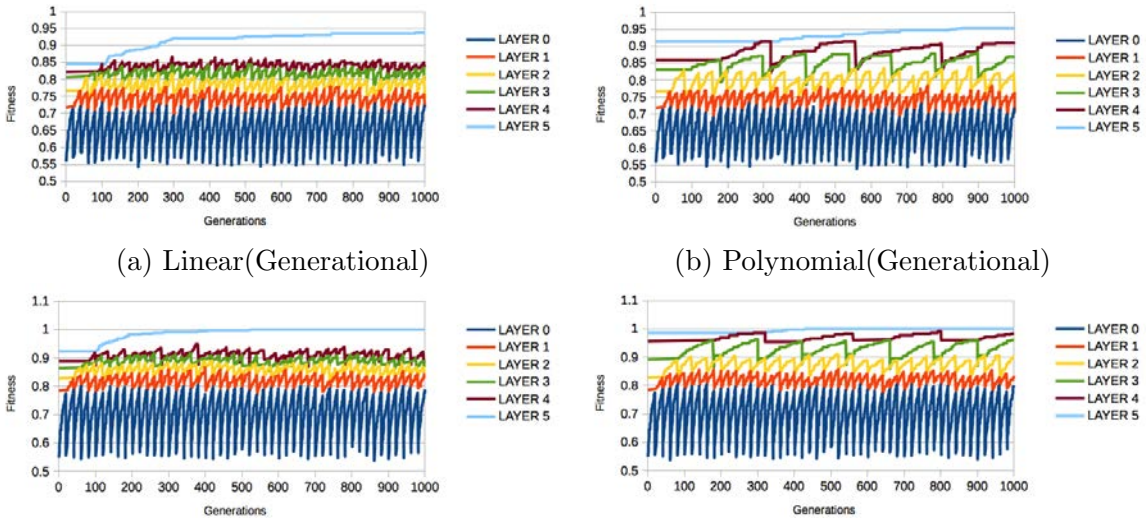
(a) ALPS GA

(b) ALPS GP

Figure A.2: (a)Comparing ALPSGA Steady State and Generational replacement strategy on a Max-Ones Problem and (b)Pima Indians Diabetes dataset using two replacement strategies

A.1.1 Ageing Scheme

Two similar ALPS configurations with varying age schemes were tested on the same dataset and the results for the linear (Figure A.3a and A.3c) and polynomial (Figure A.3b and A.3d) ageing schemes are shown in Figure A.3. Each of the aging schemes offered a unique age limit for all layers. The maximum allowed age value for a layer affects how long evolution proceeds in a particular layer and was observed to have an effect on the quality of solution. It was observed that the polynomial ageing scheme outperformed the linear ageing scheme and could be attributed to the fact that evolution progresses for a longer number of generations in lower layers before individuals age into upper layers.



(a) Linear(Generational)

(b) Polynomial(Generational)

(c) Linear(Steady State)

(d) Polynomial(Steady State)

Figure A.3: ALPSGA Steady State and Generational Replacement strategies on two Ageing Schemes

In the generational replacement strategy new offspring individuals are bred to replace the next generation of evolution while in the steady state strategy, offspring individuals compete with their ancestors (and parents) in the same population. These two strategies (see Figure A.3) do not show any significant difference in performance but exhibited some problem–dependent differences. In Figure A.3c, the steady state ALPS discovered the ideal individual and had a better training performance than generational ALPS whereas in Figure A.1 on a different problem set, generational ALPS outperformed the steady state ALPS. The choice for either strategies on an experiment is purely based on convenience therefore other major experimentations in this work used generational replacement strategy.

A.2 Diversity Enhancement

The ALPS variant has consistently shown its relevance as a diversity enhancement strategy during training. This is shown by the continuous improvement recorded even when canonical GP was stuck in local optima (see Figure A.2). Both plots in Figure A.2 only feature the last layer (layer 5) which contains the best individuals. The reader is referred to [29] for a thorough discussion on the behavior of ALPS and how it serves as a better meta-heuristic to the canonical strategy.

A.3 Feature Count and Probability Calculation

A frequency count system is started when individuals are shipped into the last layer. The Pima Indians Diabetes dataset consisted of 8 features plus an ephemeral constant. When evolution begins, the system attempts to use the default probability values for each terminal symbol and switches to evolution–based probability calculation when the last layer is started. If no default probability values are specified in the parameter file, an equal probability is assumed for each terminal symbol at the very first start of evolution in the bottom layer. This means all terminal symbols have the same chance of selection when generating tree expressions.

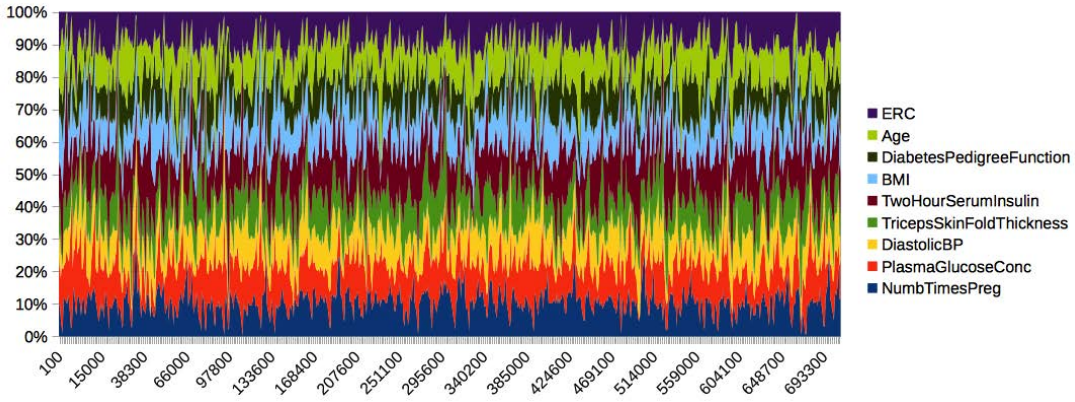


Figure A.4: Percentage contribution of each feature per layer in the FSALPS run using Pima Indians Diabetes dataset. Evolution was initialized using equal probability settings for all features

In the first test, the FSALPS system was run without performing a frequency count; this means that the system was to run using only the default probability values of terminals. Figure A.4 shows an equal selection process for the bottom layer in a complete FSALPS run. This is different from what goes on in an ALPS run where feature selection is always done randomly. Each color band in Figure A.4 represents a feature and the distribution is observed to be fairly uniform through the entire run for all bottom–layer restarts. This also means that during mutation, each random sub–tree constructed uses an equal probability in the selection of the terminal symbols.

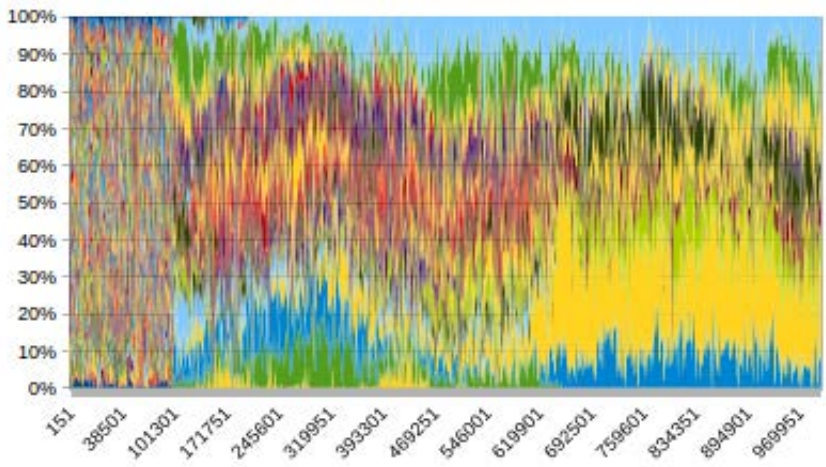
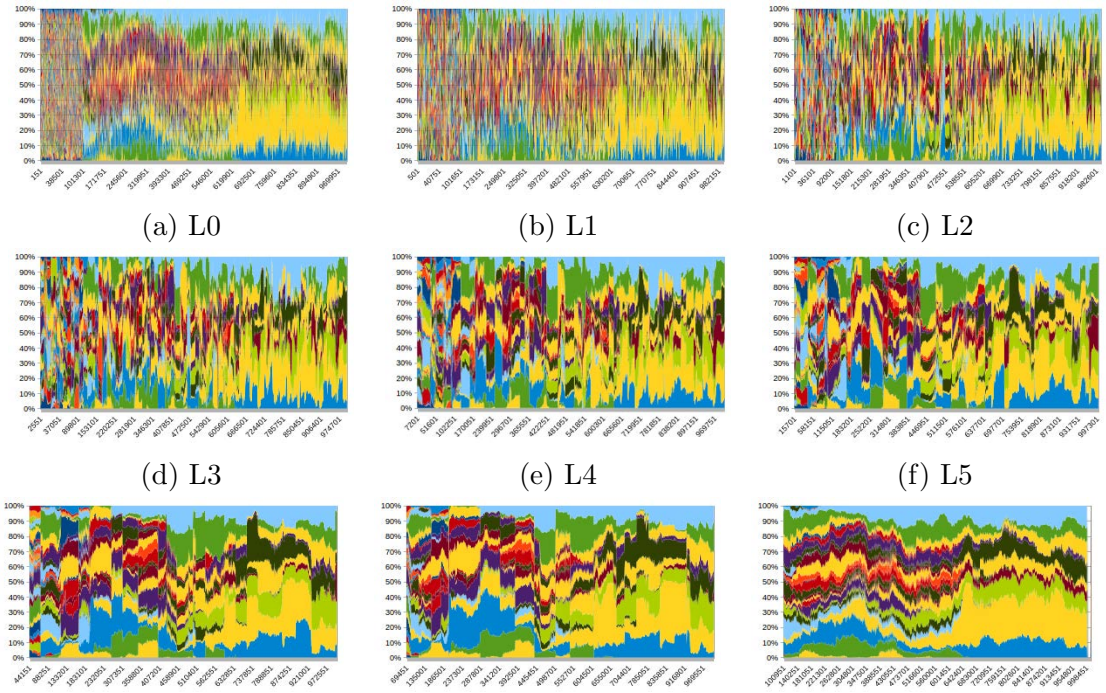


Figure A.5: Percentage contribution of each feature per layer in the ALPS run using Sonar dataset. Evolution was initialized with equal probability settings for all 60 features and performing periodic feature count and feature probability calculation

FSALPS is used to facilitate GPs default feature selection process. The directed feature selection process is done without restricting the search space to solutions within a non–promising area of the fitness landscape. At any point in the evolution process, features are not completely eliminated; but rather, they have a small chance to reappear.

In Figure A.6, probability values were computed using frequency count from the last layer. The feature selection process recorded for this approach is greedy. For instance, the 61–feature sonar dataset was reduced to 10 features in the entire population by the end of the run. Although the ideal individual was found with only 10 features (see Figure A.6i), the solution tree was not generalizable to unseen data resulting in a poor testing performance. The likely problem of over fitting due to rapid feature elimination necessitated a new approach in which feature count is performed on the entire population instead of only the last layer. It is expected that the most relevant features will have higher counts in the population. We developed a number of feature count strategies (see Section 4.3) that guarantees that less relevant features are not completely eliminated at any point in the evolution process.



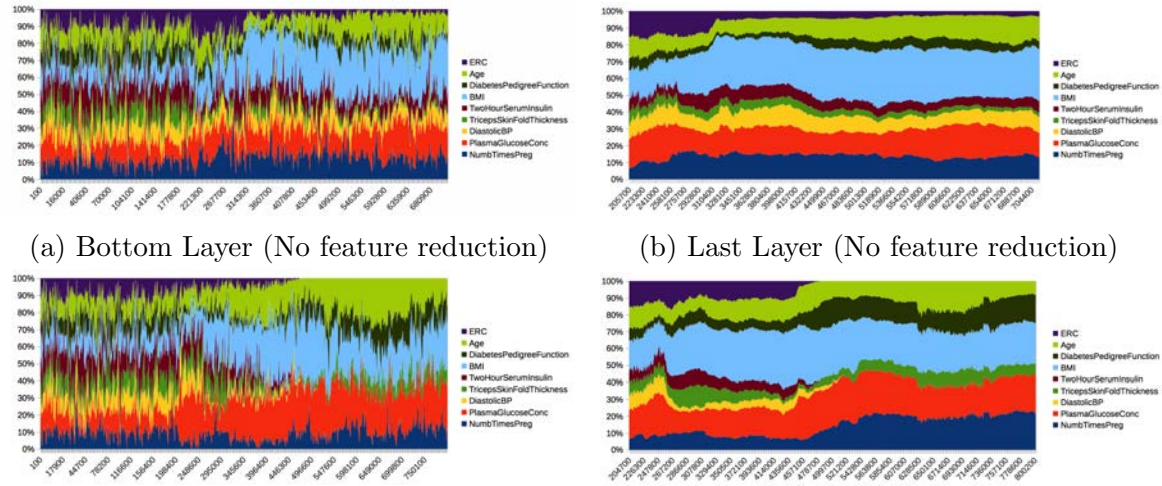
(a) L0 (b) L1 (c) L2

(d) L3 (e) L4 (f) L5

(g) L7 (h) L8 (i) L9

Figure A.6: Percentage contribution of each feature per layer in the FSALPS run using feature frequency count of only L9

In Figure A.7 (a and c) a change in the distribution is noticed around evaluation $200,000$ when individuals appeared in the last layer. In Figure A.7, two separate runs of FSALPS are shown; one shows a feature reduction from 9 to 6 while the other did not eliminate any feature. The bottom layer begins with an equal probability value for all features. The frequency count process is started immediately individuals are introduced into the last layer and for every other restart in the bottom layer. A similar frequency calculation using the entire population was done on the Sonar dataset. This time around, the feature selection process was less greedy and produced a final classifier that is more generalizable.



(a) Bottom Layer (No feature reduction)

(b) Last Layer (No feature reduction)

(c) Bottom Layer (feature reduction)

(d) Last Layer (feature reduction)

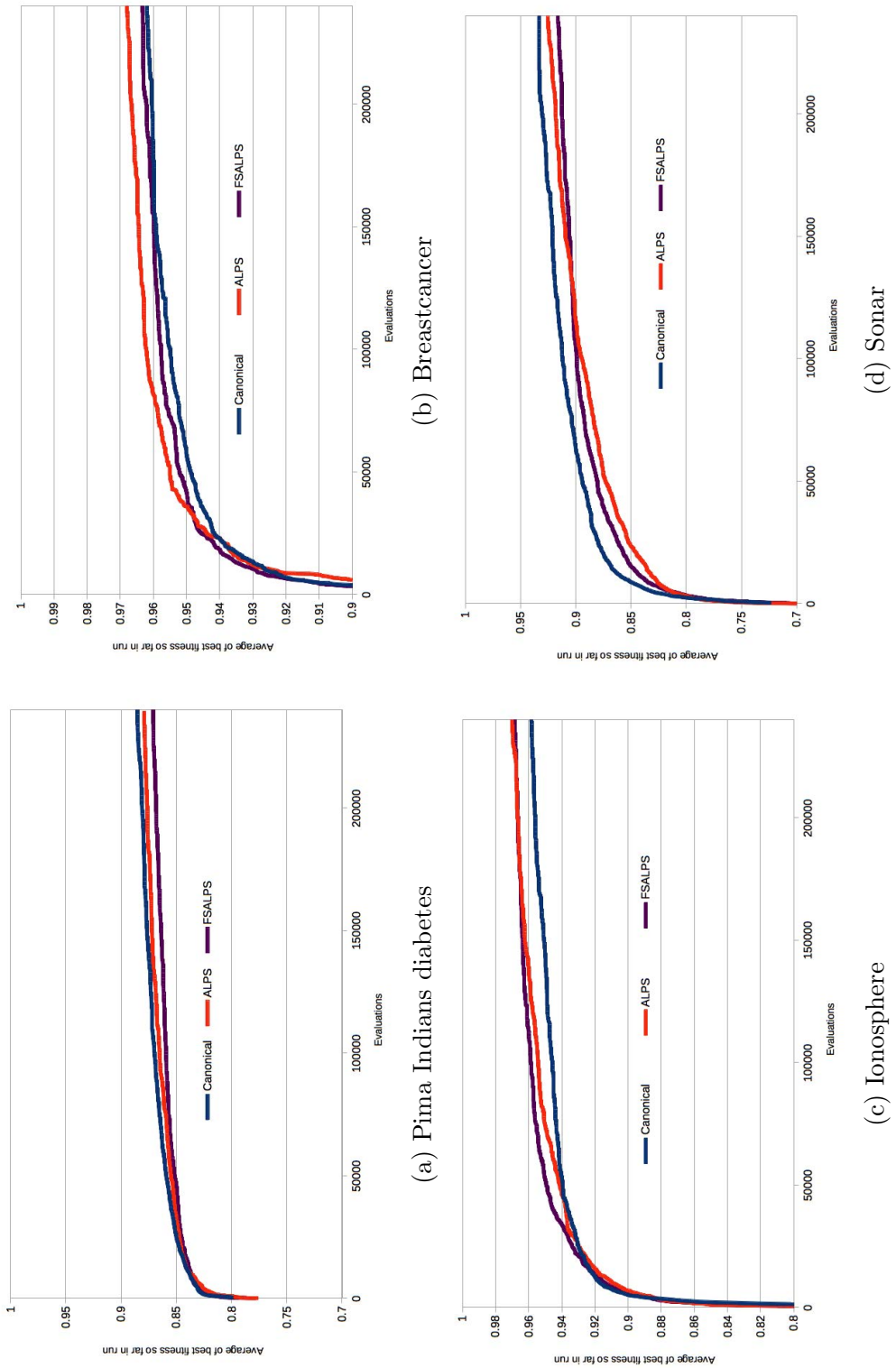
Figure A.7: Percentage contribution of each feature per layer in the FSALPS run on Pima Indians Diabetes dataset

A.4 Results

Further discussion and diagrams for the classifications experiments are discussed here.

A.4.1 Training Performance

Canonical GP, ALPS and FSALPS were run on training examples for $250,000$ evaluations. Results of training are shown in the performance plot in Figure A.8. We did not find any signs of premature convergence in all three strategies.



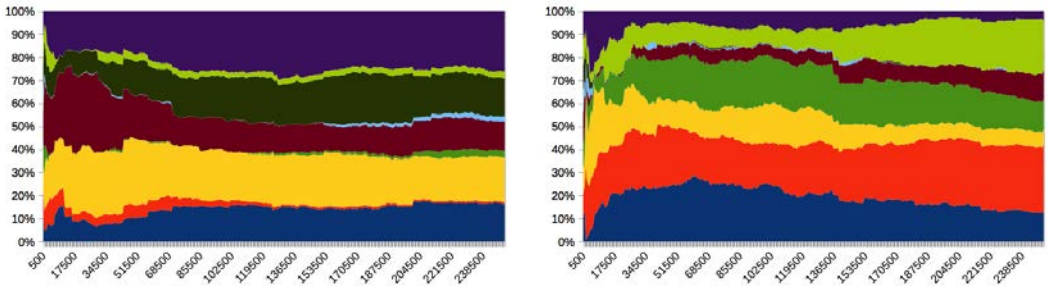
(a) Pima Indians diabetes (b) Breastcancer

(c) Ionosphere (d) Sonar

Figure A.8: Training performance plot

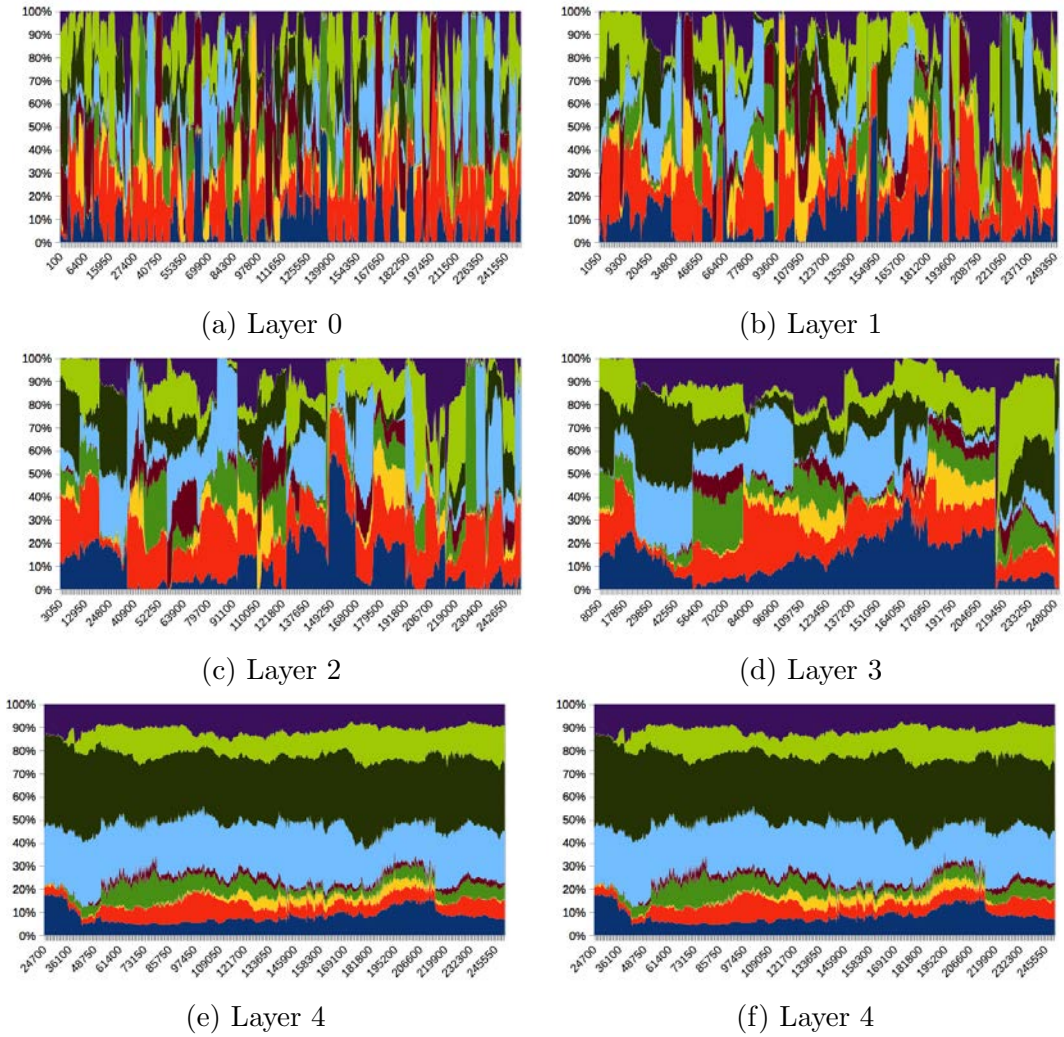
A.4.2 Feature Analysis

FSALPS emerged as a superior feature reduction technique in all 4 datasets. Out of the three strategies, FSALPS used the list features for each dataset with corresponding high classification accuracy. Figure A.12 — A.20 shows a plot of how features evolved in the population of each strategy studied. In these diagrams it becomes very obvious why canonical GP could easily land in premature convergence if longer evaluations where observed, this is due to reliance on features selected during initialization. This inherent problem in could be overcome by increasing the mutation rate in canonical GP. The problem with an increased mutation rate is "random walk", which could adversely affect evolution of good chromosomes. ALPS was the least feature selection strategy due to regular introduction of random features (see Layer 0 of ALPS plots) into the population without considering relevance of such features in the GP population.



(a) Best Individual (b) Individual with minimum features

Figure A.9: Pima: Percentage contribution of each feature in canonical GP run with (a) overall best individual had 9 features and 94.12% classification accuracy (b) minimum features had best individual with 7 features and 52.94% classification accuracy.



(a) Layer 0

(b) Layer 1

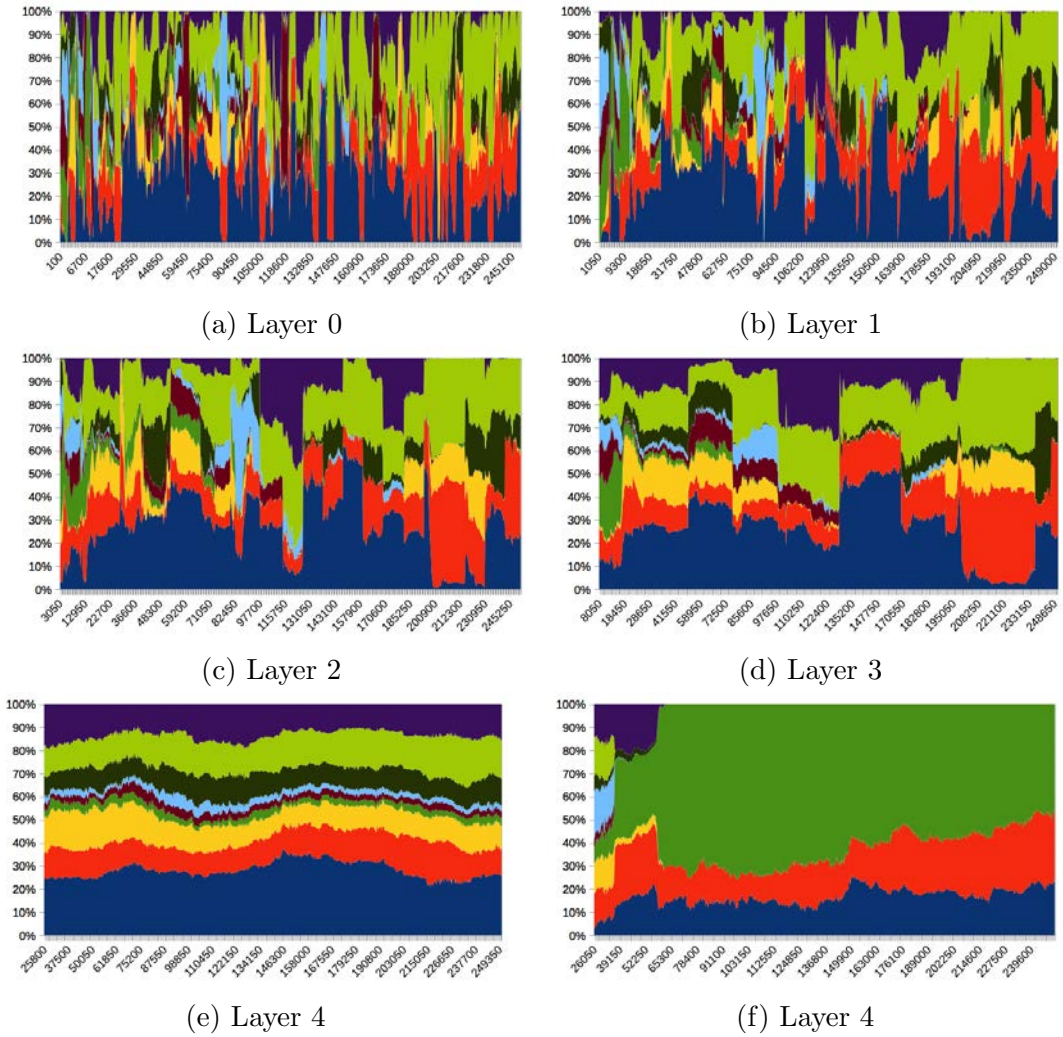
(c) Layer 2

(d) Layer 3

(e) Layer 4

(f) Layer 4

Figure A.10: Pima: Percentage contribution of each feature per layer in the ALPS run with (a–e) Layer 4 of best individual having 8 features and 88.23% classification accuracy (f) the best also individual scored the minimum number of features



(a) Layer 0

(b) Layer 1

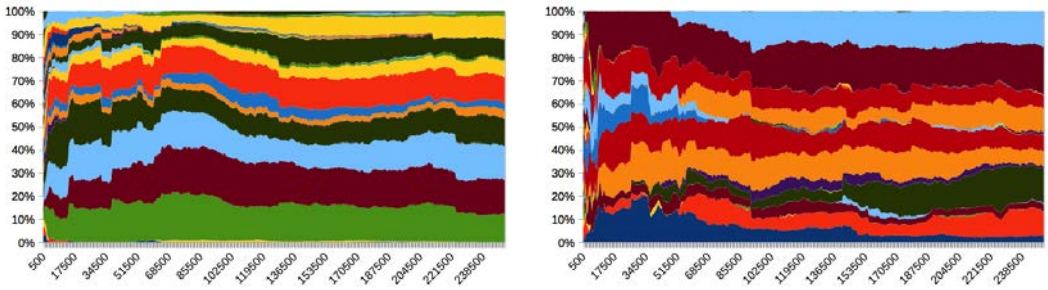
(c) Layer 2

(d) Layer 3

(e) Layer 4

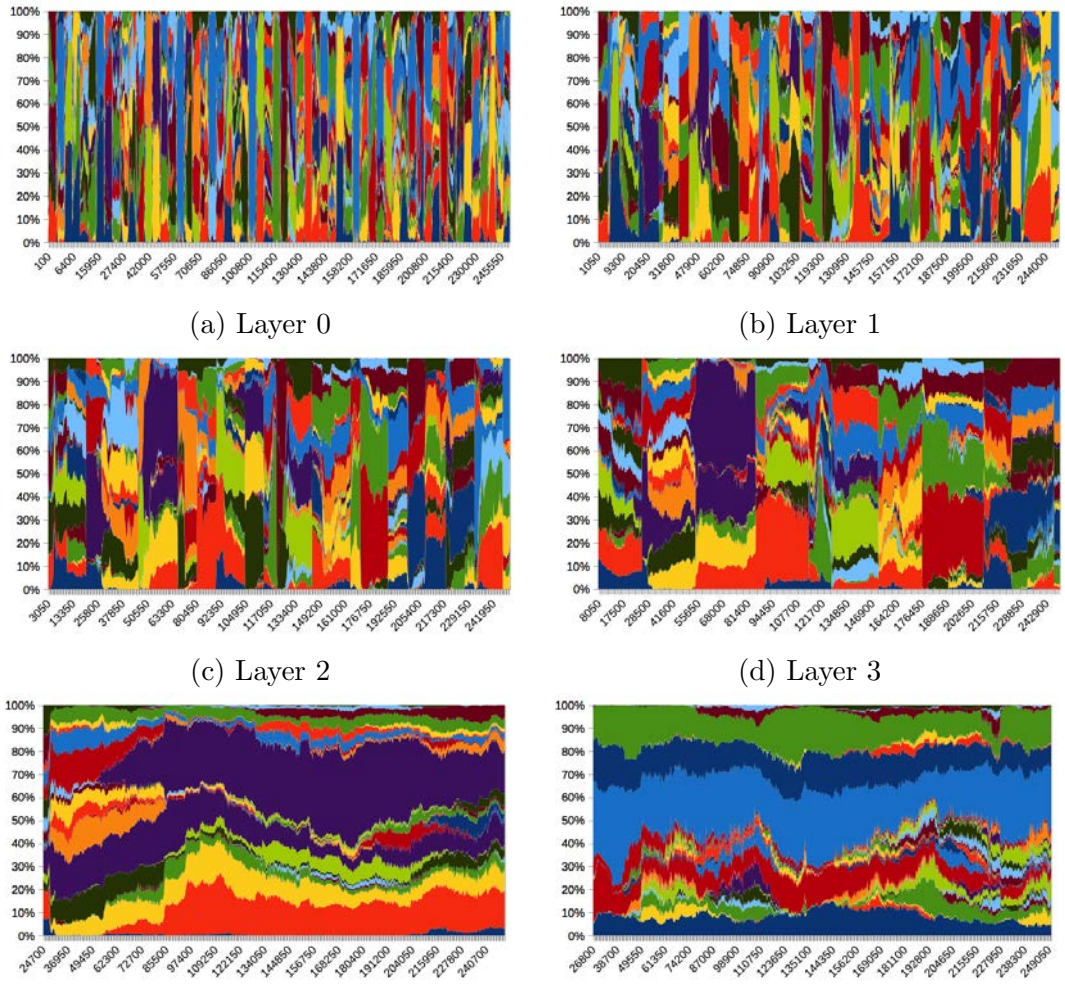
(f) Layer 4

Figure A.11: Pima: Percentage contribution of each feature per layer in the FSALPS run with (a–e) Layer 4 of best individual having 9 features and 94.12% classification accuracy (f) Layer 4 of individual with minimum features had 3 features and 75% classification accuracy.



(a) Best Individual (b) Individual with minimum features

Figure A.12: Breastcancer: Percentage contribution of each feature in canonical GP run with (a) overall best individual had 13 features and 100.00% classification accuracy (b) minimum features had best individual with 12 features and 96.43% classification accuracy.

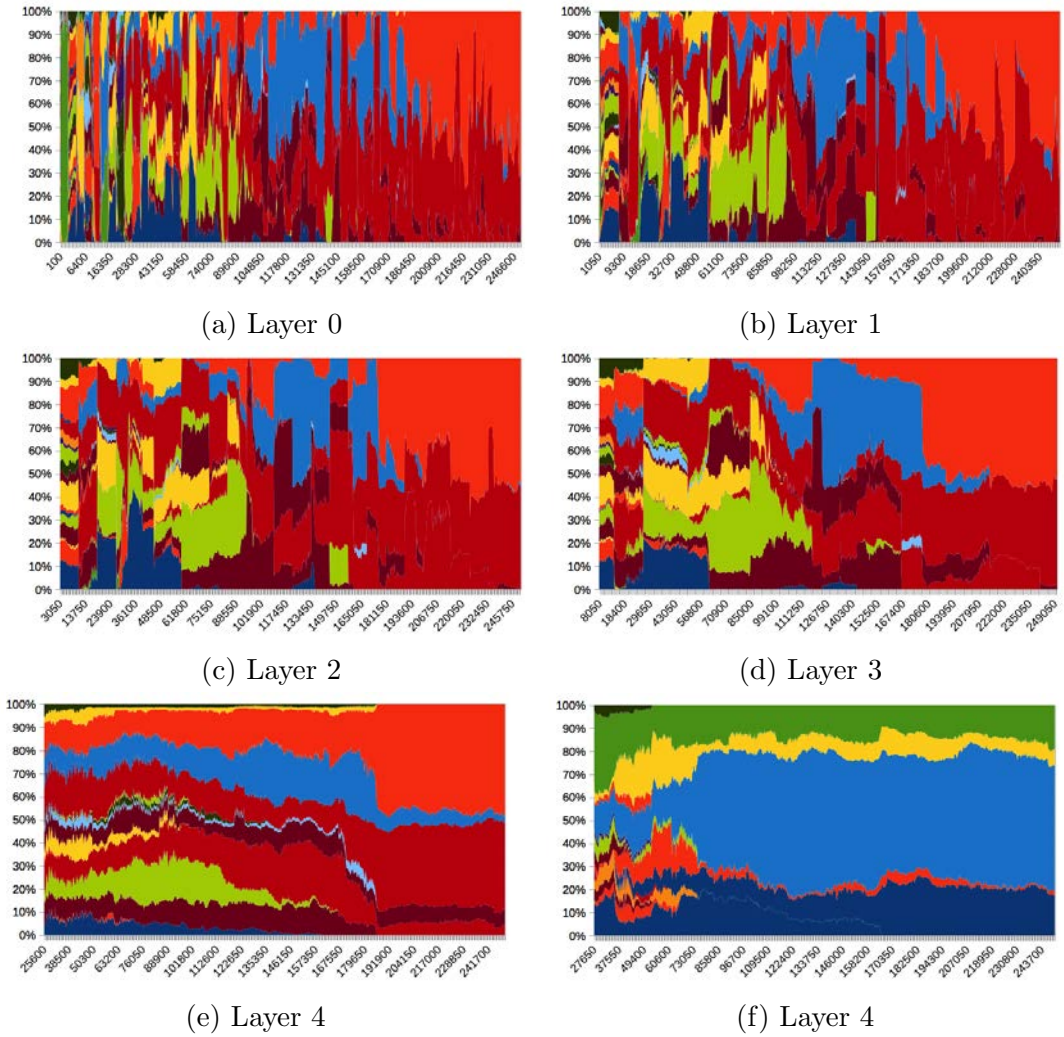


(a) Layer 0 (b) Layer 1

(c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 4

Figure A.13: Breastcancer: Percentage contribution of each feature per layer in the ALPS run with (a–e) Layer 4 of best individual having 21 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 6 features and 89.66% classification accuracy.



(a) Layer 0

(b) Layer 1

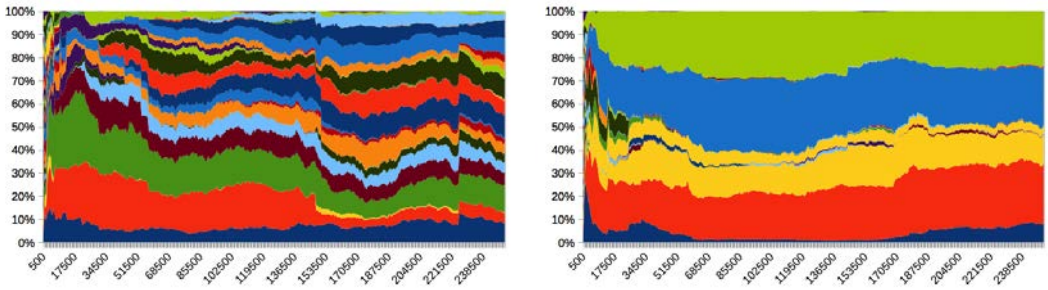
(c) Layer 2

(d) Layer 3

(e) Layer 4

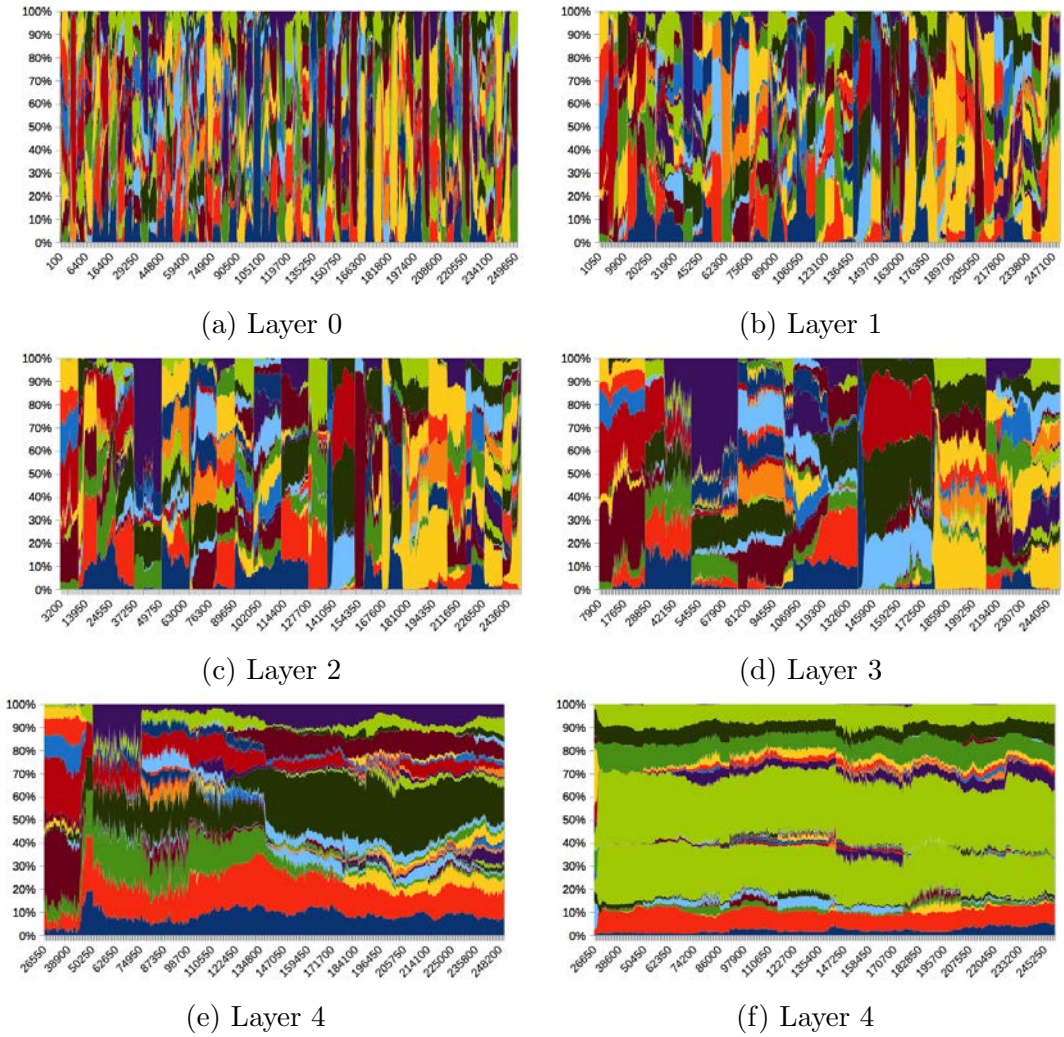
(f) Layer 4

Figure A.14: Breastcancer: Percentage contribution of each feature per layer in the FSALPS run with (a–e) Layer 4 of best individual having 6 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 5 features and 96.55% classification accuracy.



(a) Best Individual (b) Individual with minimum features

Figure A.15: Ionosphere: Percentage contribution of each feature in canonical GP run with (a) overall best individual had 14 features and 100.00% classification accuracy (b) minimum features had best individual with 9 features and 77.78% classification accuracy.



(a) Layer 0

(b) Layer 1

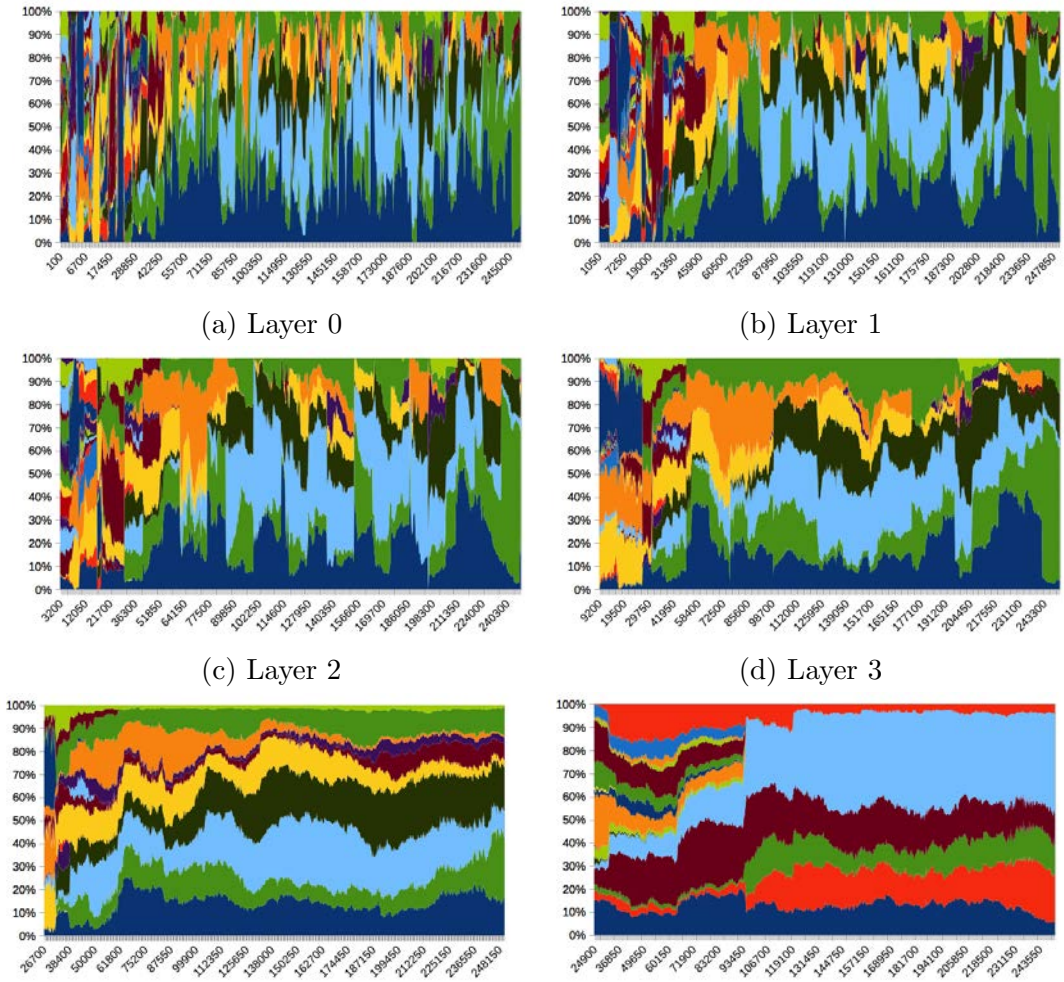
(c) Layer 2

(d) Layer 3

(e) Layer 4

(f) Layer 4

Figure A.16: Ionosphere: Percentage contribution of each feature per layer in the ALPS run with (a–e) Layer 4 of best individual having 19 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 13 features and 88.24% classification accuracy.



(a) Layer 0

(b) Layer 1

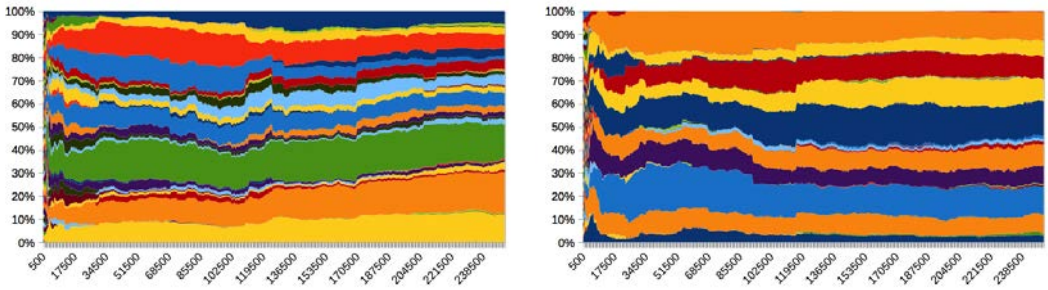
(c) Layer 2

(d) Layer 3

(e) Layer 4

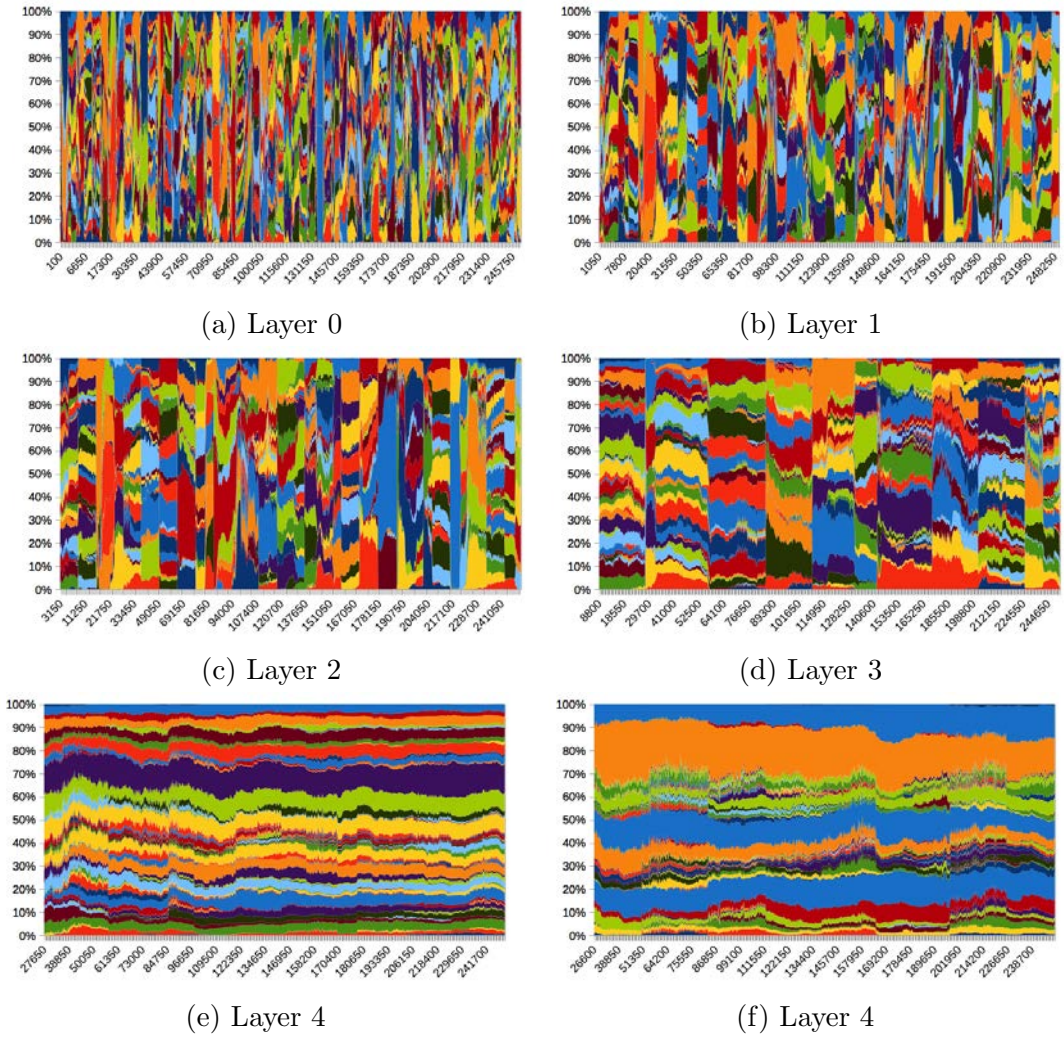
(f) Layer 4

Figure A.17: Ionosphere: Percentage contribution of each feature per layer in the FSALPS run with (a–e) Layer 4 of best individual having 11 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 7 features and 88.89% classification accuracy.



(a) Best Individual (b) Individual with minimum features

Figure A.18: Sonar: Percentage contribution of each feature in canonical GP run with (a) overall best individual had 27 features and 100.00% classification accuracy (b) minimum features had best individual with 16 features and 70.00% classification accuracy.



(a) Layer 0

(b) Layer 1

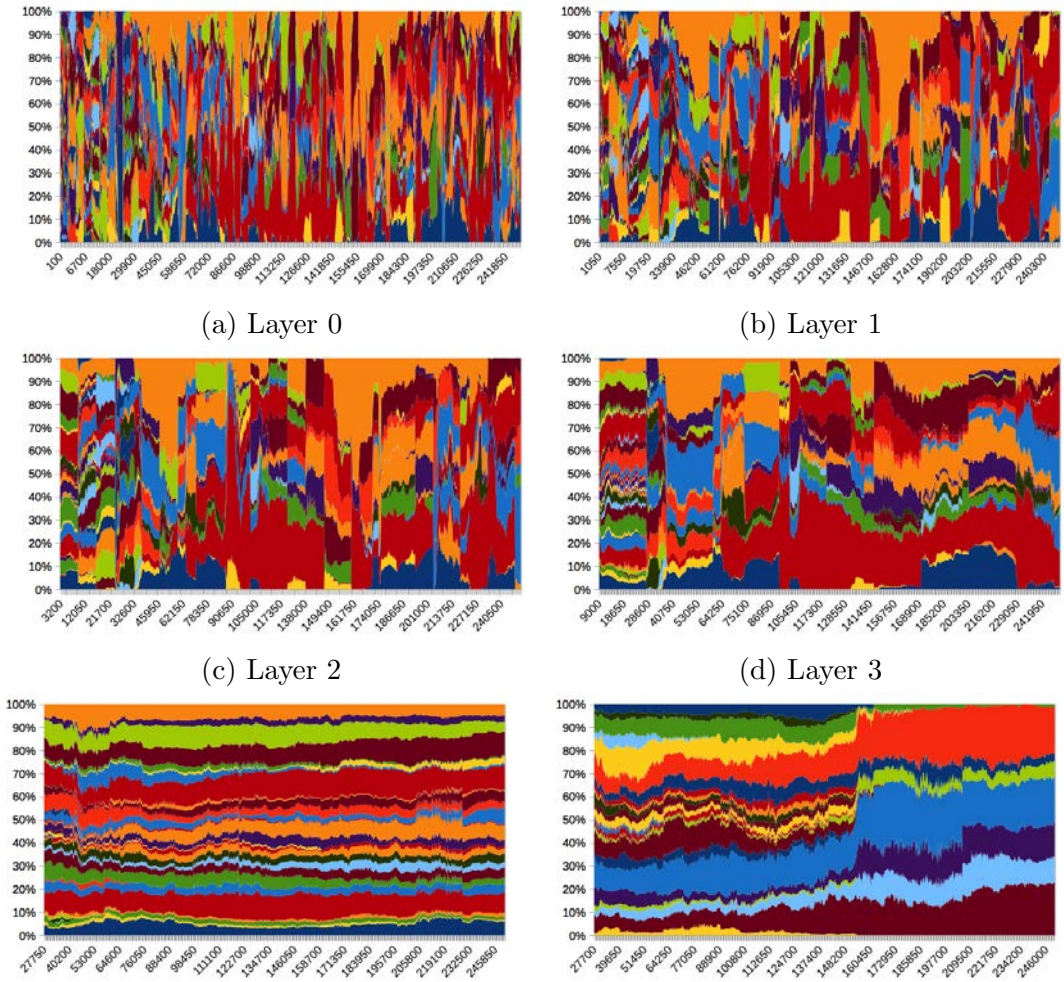
(c) Layer 2

(d) Layer 3

(e) Layer 4

(f) Layer 4

Figure A.19: Sonar: Percentage contribution of each feature per layer in the ALPS run with (a–e) Layer 4 of best individual having 45 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 28 features and 70% classification accuracy.



(a) Layer 0 (b) Layer 1

(c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 4

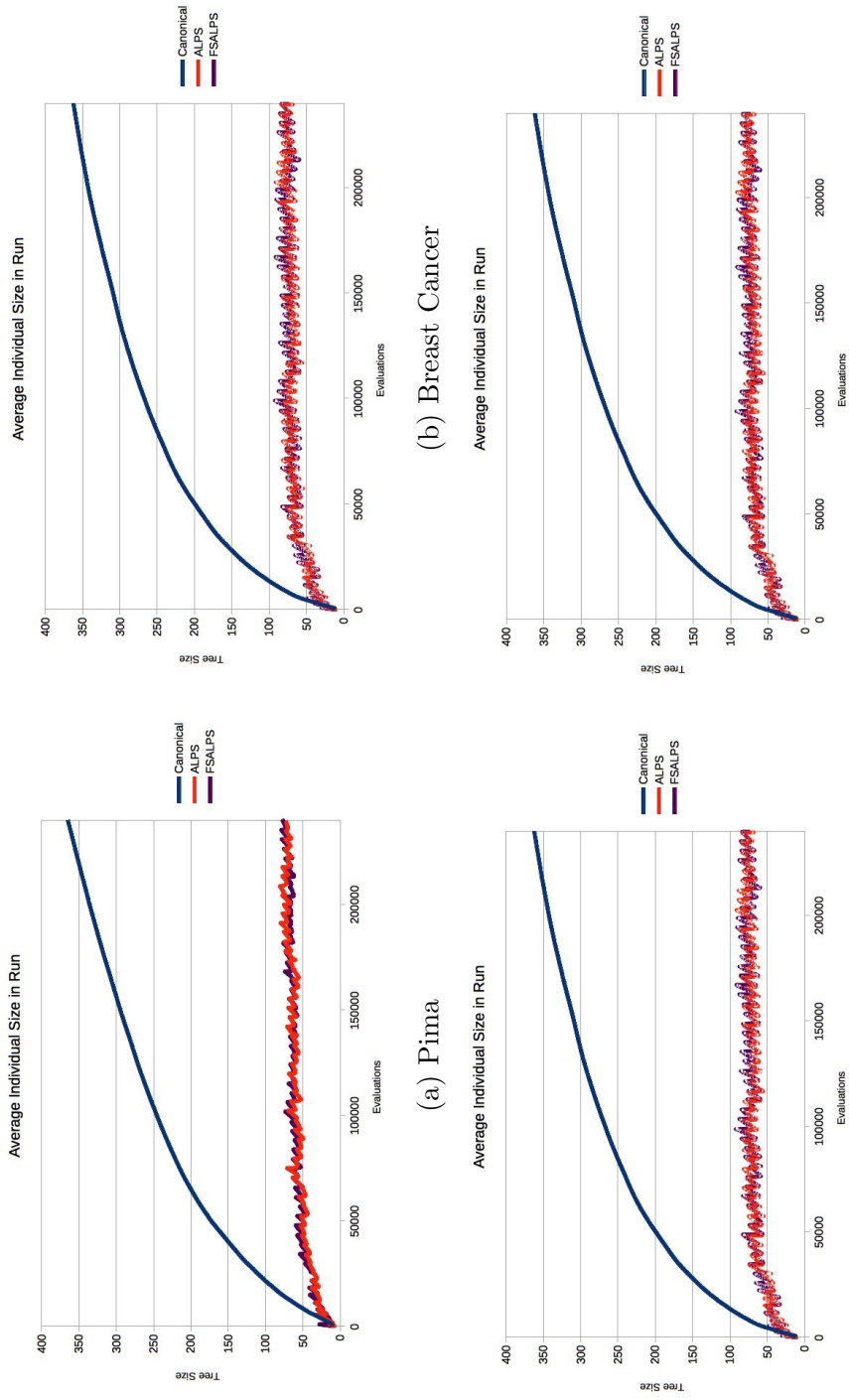
Figure A.20: Sonar: Percentage contribution of each feature per layer in the FSALPS run with (a–e) Layer 4 of best individual having 27 features and 100.00% classification accuracy (f) Layer 4 of individual with minimum features had 7 features and 80.00% classification accuracy.

A.4.3 Performance Analysis

In Figure A.21, we look at the growth of tree size per run for each experiment. The results were consistent for all datasets. ALPS and FSALPS consistently produced smaller and more efficient trees when compared to canonical GP. The growth in tree size of the best individual per run (see Figure A.23) or per generation (see Figure A.22) in canonical GP does not happen at a faster rate as growth in average tree size per run in canonical GP. The rapid growth in tree–size in Figure A.21 shows evidence of bloat (large but ineffective individuals).

The space and time complexity for all three algorithms are measured by plotting

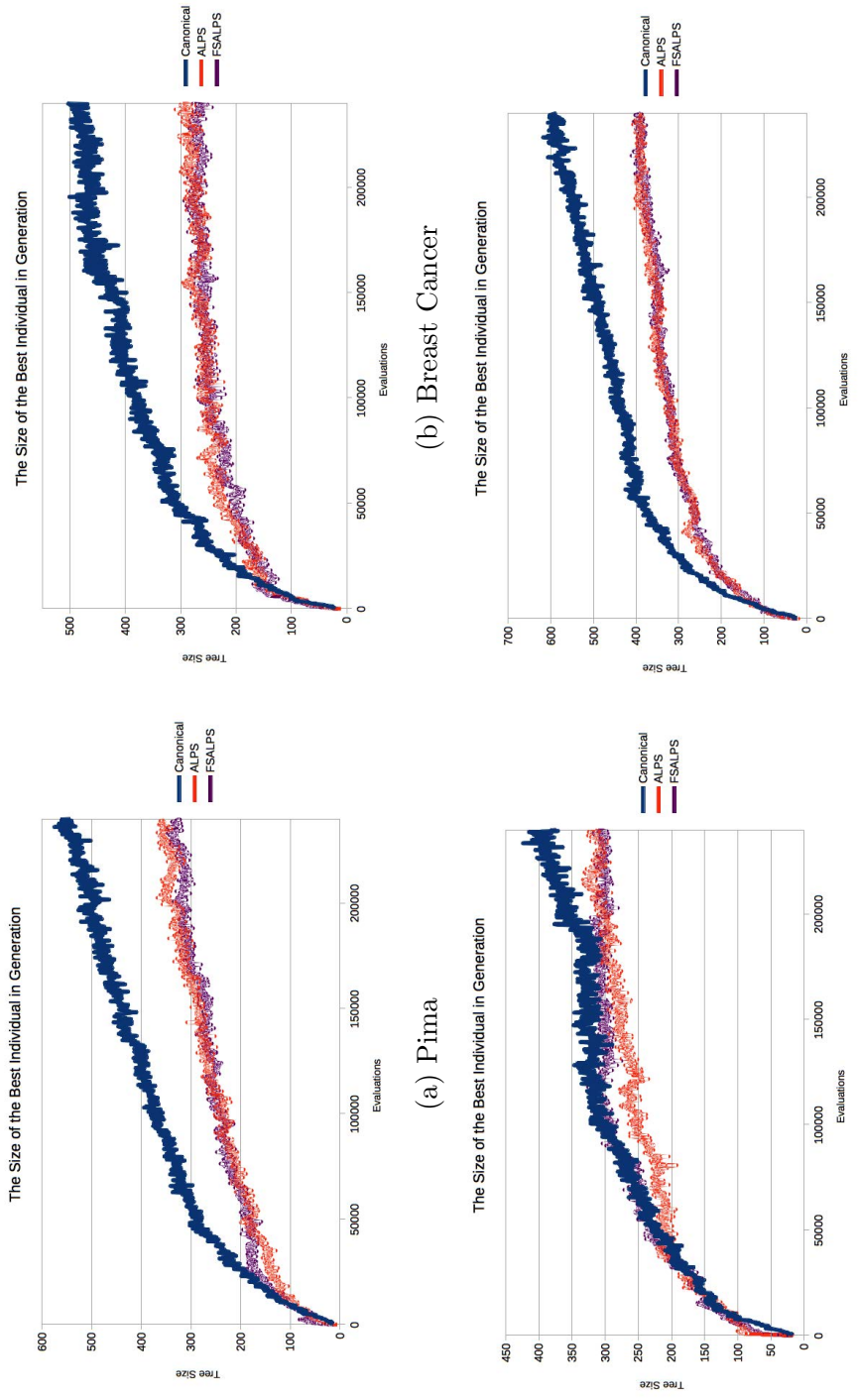
a logarithmic plot of the time and tree size used for each strategy. In Figure A.24 and Figure A.25 the rate of growth of canonical GP is exponential and approximates to the order of growth of $O(n^2)$) when compared to ALPS and FSALPS that are approximately between $O(n)$ and $O(nlog(n))$. The time and memory stabilization achieved by ALPS and FSALPS makes them resource efficient and are more suitable in handling large problems that will usually require huge resources.



(a) Pima (b) Breast Cancer

(c) Ionosphere (d) Sonar

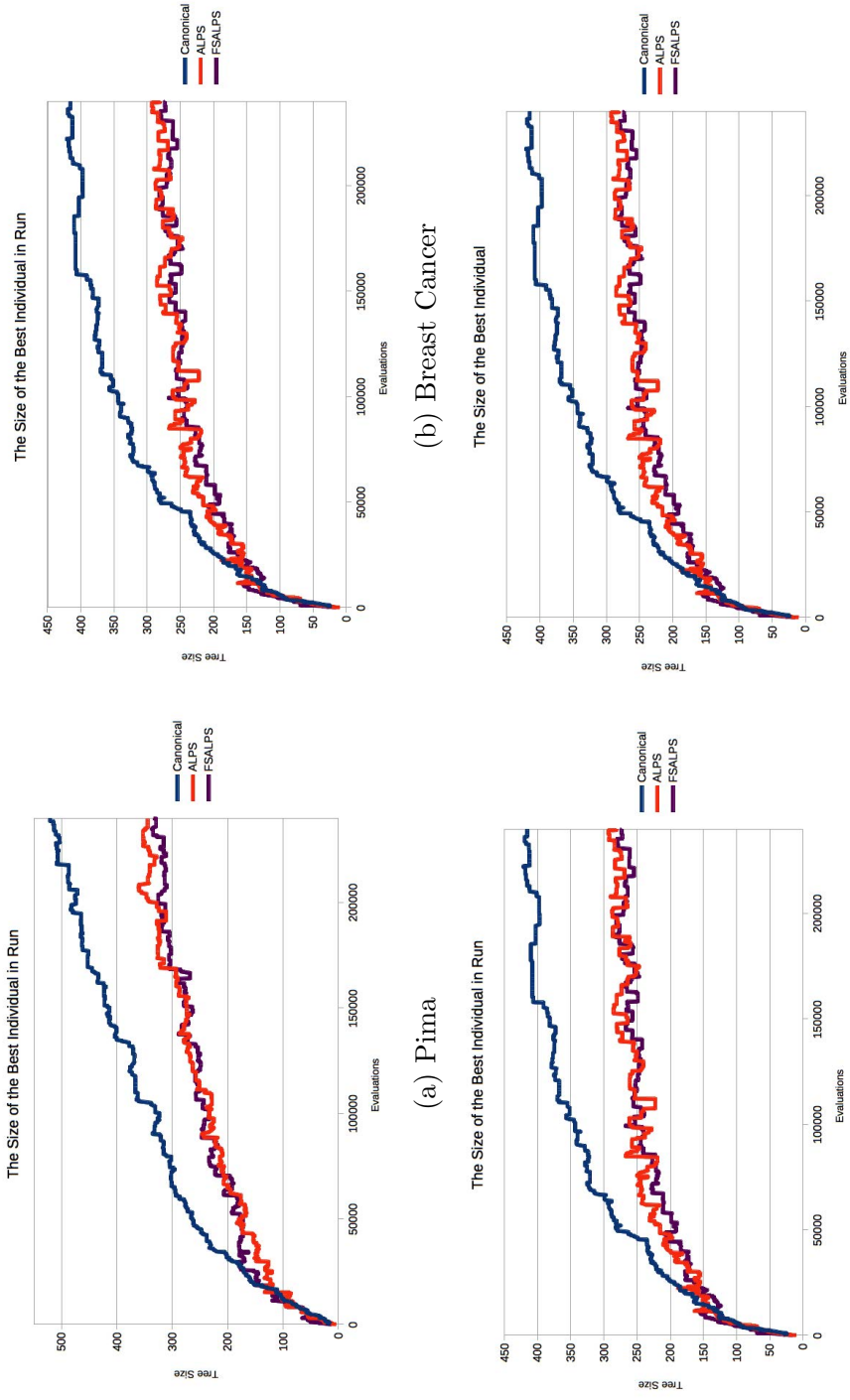
Figure A.21: Average individual size per run for 250,000 evaluations



(a) Pima (b) Breast Cancer

(c) Ionosphere (d) Sonar

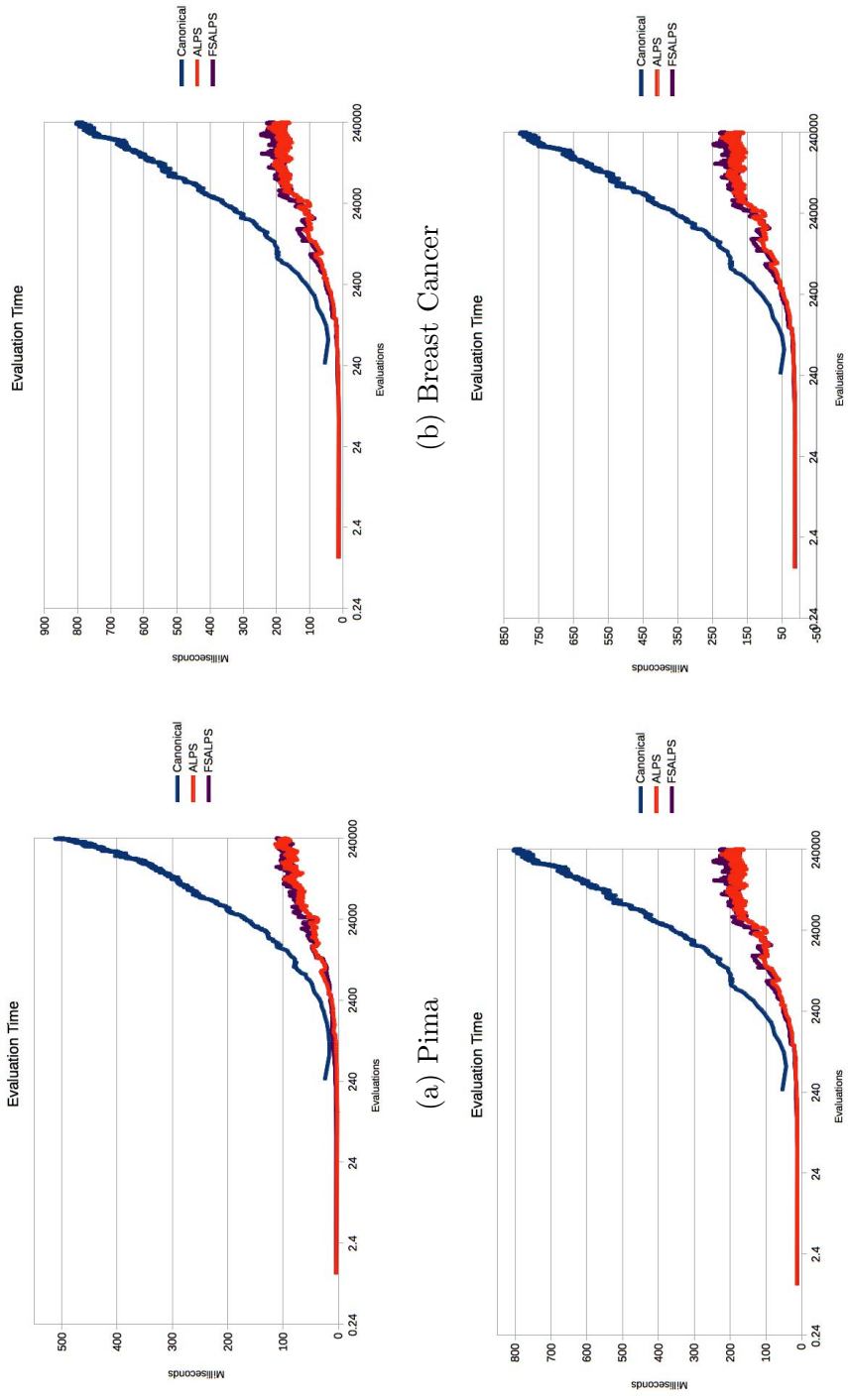
Figure A.22: Size of best individual per generation for 250,000 evaluations



(a) Pima (b) Breast Cancer

(c) Ionosphere (d) Sonar

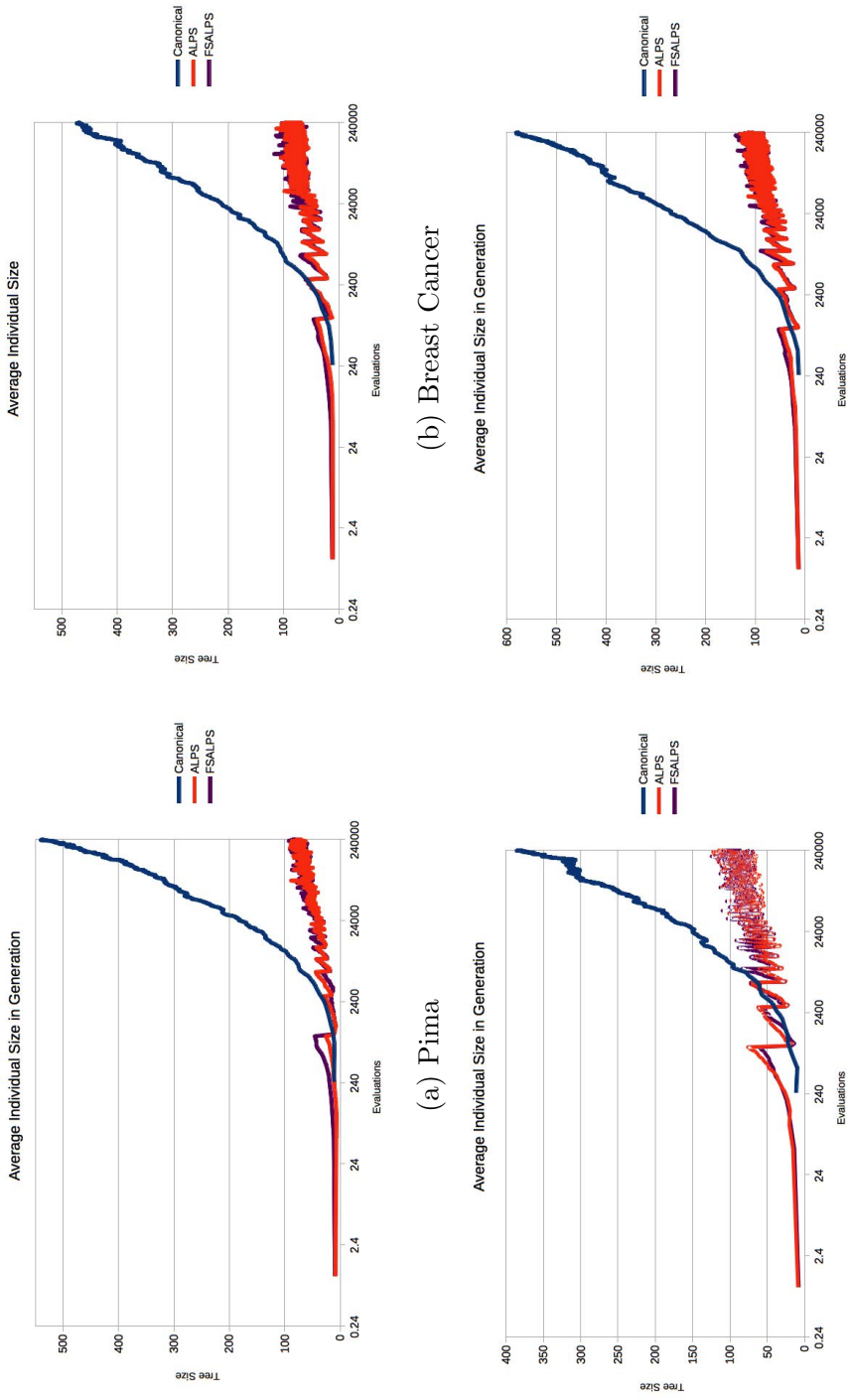
Figure A.23: Size of best individual per run for $250,000$ evaluations



(a) Pima (b) Breast Cancer

(c) Ionosphere (d) Sonar

Figure A.24: Size of best individual per generation for 250,000 evaluations



(a) Pima (b) Breast Cancer

(c) Ionosphere (d) Sonar

Figure A.25: ALPSGA Steady State and Generational Replacement strategies on two Aging Schemes

A.4.4 Solution Trees

Due to the huge size of tree expressions (especially for canonical), we included very few of the best solution trees for each of the strategies and mostly excluded results of canonical GP. The LISP S-Expressions are reflective of the tree size plots shown in Figure A.21d — A.23.

(- (% (+ (+ (- (% (- (% (max wdbc24 wdbc27) wdbc27) (+ (max wdbc24 wdbc27) (+ (max wdbc24 wdbc27) wdbc24))) wdbc27) (+ wdbc24 (% (+ (* wdbc24 wdbc12) (min wdbc27 wdbc18)) wdbc27))) (- (% (+ (+ (+ (max wdbc24 wdbc27) (% (- (% (+ wdbc24 (min (max wdbc24 wdbc27) wdbc18)) wdbc27) wdbc25) wdbc27)) (% (log10 wdbc24) wdbc27)) (min wdbc25 wdbc18)) wdbc27) (+ (max wdbc24 (% (- (% wdbc27 wdbc24) (+ (+ (max wdbc24 (% (- (% (max wdbc24 wdbc27) wdbc27) (% (max wdbc24 wdbc27) wdbc27)) wdbc27)) wdbc24) wdbc24)) wdbc27)) wdbc24))) (% (- (% (+ (max wdbc24 (% (max wdbc24 wdbc27) wdbc27)) (min (max wdbc24 wdbc27) wdbc18)) wdbc27) (+ (+ (min (max (% (- (min (% (% wdbc24 wdbc27) (log10 wdbc24)) (* wdbc27 wdbc27)) (max (log10 wdbc24) (ln wdbc24))) wdbc27) (max wdbc24 wdbc27)) wdbc18) (- (% (- (% (+ (max wdbc24 wdbc18) wdbc27) wdbc27) (+ (max wdbc24 wdbc24) (+ (max wdbc24 wdbc27) wdbc24))) wdbc27) (+ (max wdbc24 wdbc24) (% wdbc24 wdbc27)))) (- (% (- (% (+ (max wdbc24 wdbc27) wdbc27) wdbc27) (+ (max wdbc24 wdbc27) (+ (max wdbc24 wdbc27) wdbc24))) wdbc27) (+ (+ (max (max wdbc24 wdbc27) wdbc27) (- (% (% wdbc12 wdbc27) wdbc27) (+ (max wdbc24 (% (- (% (% wdbc12 wdbc27) wdbc27) (% (min wdbc12 wdbc24) wdbc27)) wdbc27)) wdbc24))) (% (+ (max wdbc24 wdbc27) (min wdbc25 wdbc18)) wdbc27))))) wdbc27)) wdbc27) (+ (max (max wdbc24 wdbc27) (% (+ (max wdbc24 (% wdbc24 wdbc27)) (min wdbc25 wdbc18)) wdbc27)) wdbc24))

Code 1: LISP S-Expression best testing FSALPS individual for Breastcancer dataset with classification accuracy 100% and contains 6 unique features out of 33

(- (- (* wdbc4 (* wdbc4 (* (- (* wdbc4 (* wdbc4 (* wdbc4 wdbc22))) (* (* (max wdbc3 wdbc10) wdbc22) (- (- (- (* (* (* wdbc22 wdbc22) wdbc22) wdbc22) (max (max (% (max (max wdbc22 wdbc22) (% wdbc5 wdbc10)) wdbc10) wdbc22) wdbc22)) (min wdbc21 wdbc22)) (* (* (- (- (% (ln wdbc3) wdbc29) (max wdbc3 (ln wdbc8))) (ln (- (% (ln wdbc22) wdbc29) (max wdbc3 (ln wdbc3))))) wdbc4) wdbc22)))) wdbc22)))

(* (- (max (- (- (- (max (% wdbc22 wdbc9) (% (max wdbc3 wdbc10) (- wdbc17 (* (ln wdbc30) (- wdbc12 (* (% (ln wdbc3) wdbc29) wdbc9)))))) (+ (- (- (% (max wdbc22 (% (max (max wdbc22 wdbc22) wdbc22) wdbc22) (* (max wdbc3 wdbc10) wdbc22))) wdbc22) wdbc3) (- (- wdbc23 (* wdbc17 wdbc22)) (% (% wdbc5 wdbc10) (- (% (ln wdbc3) wdbc29) (max wdbc3 (ln wdbc8)))))) (log10 (* (* (- (ln wdbc12) (ln wdbc3)) wdbc22) wdbc22)))) (max wdbc3 wdbc10)) (% (* (* (max wdbc3 wdbc10) wdbc22) (- (max (max wdbc22 wdbc22) (% wdbc5 wdbc10)) (* (* (% (max wdbc22 (% (max (max wdbc22 wdbc22) wdbc22) (* (max wdbc3 wdbc10) wdbc22))) wdbc22) wdbc4) wdbc22))) (% (max wdbc3 wdbc10) (- wdbc17 (* (ln wdbc30) (- wdbc12 (* (+ wdbc29 wdbc30) wdbc9)))))))) (- (max (% (max wdbc3 wdbc10) (% wdbc8 (% (* wdbc4 (% (max wdbc22 (ln wdbc3)) wdbc22)) wdbc10))) (max (% wdbc27 wdbc9) (* (- (- (- wdbc12 (* (ln wdbc24) (% (ln wdbc3) wdbc29))) (* wdbc28 (ln wdbc12))) (* wdbc28 (ln wdbc30))) wdbc25))) wdbc3)) wdbc22) (- (- (ln wdbc3) (* (* (% wdbc5 wdbc10) wdbc22) wdbc22)) wdbc22))) (* (* (- (* (ln wdbc8) (ln wdbc2)) (* (* (- (- (- (* (* (* wdbc22 wdbc22) wdbc22) wdbc22) (max wdbc3 wdbc10)) (min wdbc21 wdbc6)) (* (- (% (max (max (- (ln wdbc12) (* -0.817987 (ln wdbc12))) wdbc22) wdbc22) wdbc9) (% wdbc22 wdbc9)) wdbc22)) wdbc22) (- wdbc17 wdbc27))) wdbc22) wdbc3))

Code 2: LISP S-Expression best testing ALPS individual for Breastcancer dataset with classification accuracy 100% and contains 21 unique features out of 33

(+ (* (max (min (min (min wdbc6 (ln (log10 wdbc13))) (* (max (- wdbc8 wdbc7) wdbc8) (* wdbc15 (- (+ (- (max (log10 wdbc29) (* wdbc15 wdbc28)) (log10 wdbc6)) wdbc28) wdbc7)))) (* (* wdbc20 (* (log10 wdbc6) (- (% wdbc16 (* wdbc15 (- wdbc8 wdbc7))) (min (* wdbc6 (* wdbc6 wdbc5)) (log10 wdbc6))))) (min (- (% (+ wdbc7 wdbc28) wdbc5) (% (* (- wdbc15 (- wdbc8 wdbc7)) wdbc5) (+ wdbc28 wdbc20))) (min wdbc7 (* wdbc6 (* (log10 (+ wdbc7 (- (% wdbc16 (* wdbc6 wdbc5)) (min (+ (% wdbc16 wdbc8) wdbc20) (log10 wdbc29))))) (* wdbc15 (min (- (min (- (max (log10 wdbc6) (log10 wdbc6)) (min (* wdbc6 wdbc5) (* wdbc6 wdbc5))) (min (- (+ wdbc28 wdbc20) (+ wdbc6 wdbc20)) (log10 wdbc29))) (- wdbc8 (log10 wdbc7))) wdbc11)))))))) (max (min (max (% wdbc16 (log10 wdbc13)) (* wdbc15 wdbc28)) wdbc6) (- wdbc15 (- wdbc8 wdbc7)))) (min (min (- (+ (* (- (min (- (% wdbc16 wdbc5) (min (log10 (+ wdbc7 wdbc5)) (* wdbc6 wdbc5))) (+ wdbc28 wdbc20)) wdbc15) (min (* (log10 (* wdbc15 (- wdbc8 wdbc7))) (min (* wdbc6 wdbc5) wdbc5)) (* wdbc11 (min (% (- (+ wdbc28 wdbc20) (log10 wdbc6)) wdbc20) (* (max (log10

wdbc29) (* wdbc15 wdbc28)) (* wdbc6 wdbc5)))))) (% (* (* wdbc11 (min (% (- wdbc8 (- wdbc8 wdbc7)) wdbc20) (* (* wdbc15 wdbc28) (+ wdbc28 wdbc20)))) wdbc5) (+ wdbc28 wdbc20))) wdbc7) (min (min (% wdbc16 (% wdbc16 (+ wdbc7 wdbc5))) wdbc8) (* (max (* (log10 (* wdbc15 wdbc17)) (log10 wdbc11)) (* wdbc15 (- wdbc8 wdbc7))) (min (- (min (- (- (% (+ wdbc7 wdbc28) wdbc5) (ln (+ wdbc7 wdbc5))) (* wdbc6 wdbc5)) 0.5849673) wdbc15) (* (* wdbc15 wdbc28) (% wdbc16 (log10 wdbc13)))))))) (min (* (max (* (* (log10 (- (- wdbc8 wdbc7) wdbc7)) (- (% wdbc16 (* wdbc15 (- wdbc8 wdbc7))) (min (+ wdbc28 wdbc20) (log10 wdbc6)))) (min (- (+ wdbc28 wdbc20) wdbc15) (* wdbc11 (min (max (* wdbc8 (log10 wdbc11)) (- (min (log10 (+ (- wdbc8 wdbc7) (log10 wdbc6))) (* (- wdbc8 (log10 wdbc6)) (log10 wdbc11))) (min (log10 (+ wdbc7 wdbc5)) wdbc16))) (* (* (max (log10 wdbc6) (log10 wdbc6)) (- (ln (* wdbc15 (- wdbc8 wdbc7))) (ln (- wdbc8 wdbc7)))) (min (* wdbc6 wdbc5) (* (* wdbc15 wdbc28) (+ wdbc28 wdbc20)))))))) (ln wdbc5)) (min (% wdbc16 (log10 wdbc13)) (log10 (- (- wdbc8 wdbc7) wdbc7)))) (* (max (* (* (log10 wdbc6) (- (% wdbc16 (* wdbc15 (- wdbc8 wdbc7))) (min (* wdbc6 (* wdbc15 (- wdbc8 wdbc7))) (log10 wdbc6)))) (min (* (log10 (log10 wdbc6)) (log10 (+ wdbc28 wdbc20))) (* wdbc11 (min (% (min (* wdbc15 (- (log10 wdbc13) wdbc7)) wdbc15) wdbc20) (* (max (% wdbc16 (log10 wdbc13)) (* wdbc15 wdbc28)) (min (- wdbc8 (* wdbc6 wdbc5)) (- wdbc8 (log10 wdbc6))))))))) (* (log10 (+ wdbc7 (- (% wdbc16 (* wdbc6 wdbc5)) (min (+ (% wdbc16 wdbc8) wdbc20) (log10 wdbc29))))) (* wdbc15 (min (- (min (- (+ (- wdbc8 wdbc7) (log10 wdbc6)) (min (* wdbc6 wdbc5) (* wdbc6 wdbc5))) (min (- (min (* wdbc6 (* wdbc6 wdbc5)) (* wdbc6 wdbc5)) (+ wdbc6 wdbc20)) wdbc15)) (- wdbc8 (log10 wdbc7))) (log10 (* wdbc15 wdbc28)))))) (min (% wdbc16 (log10 (min (% wdbc16 (log10 wdbc13)) (log10 (- (- wdbc8 wdbc7) wdbc7))))) (log10 (- (- wdbc8 wdbc7) wdbc7))))))) (- (+ (% (* (- wdbc8 (log10 wdbc6)) wdbc5) (+ wdbc28 wdbc20)) (log10 wdbc6)) wdbc5))

Code 3: LISP S-Expression best testing canonical GP individual for Breastcancer dataset with classification accuracy 100% and contains 13 unique features out of 33

(max (log10 (* (max (max ion21 (min ion28 ion7)) (+ (+ (max (ln ion7) (max (ln ion7) ion6)) (max (- ion7 ion15) ion6)) (% (max (+ ion7 (min ion28 ion21)) (ln ion17)) ion28))) (% (* (+ ion7 ion32) (% (log10 (% (log10 (* ion28 (+ ion1 (* (ln (- ion6 ion6)) ion6)))) (max (min ion1 ion1) (+ ion4 ion7)))) (ln (max (+ ion7 (ln ion6)) (+ (+ (max (ln ion7) ion6) (ln ion6)) (ln ion15)))))) (ln (max (max (ln (max

(min ion28 ion21) (+ (max (+ ion7 (min ion28 ion21)) (ln ion17)) (ln ion15)))) (ln ion15)) (max (ln (max (+ ion7 (ln ion1)) (+ ion4 (max ion7 (ln ion17))))) (ln (max (- ion7 ion15) (+ ion7 (- (max (min ion1 ion1) (+ ion4 ion7)) ion6)))))))))) (+ ion7 (+ ion22 (min (- (% (* (+ ion7 ion32) (% (ln ion15) (ln (max (max ion21 (min ion28 ion7)) (min ion28 ion7))))) (log10 (% (max (% (log10 (% (% (- (+ ion6 ion7) ion7) (% ion1 ion28)) ion4)) (% ion1 ion28)) (max (log10 (* ion28 (+ ion1 (* (ln (- ion6 ion6)) ion6)))) ion6)) (ln ion28)))) (max (max (log10 (log10 (min ion28 ion7))) (log10 ion15)) (ln ion6))) (min ion6 ion28)))))

Code 4: LISP S-Expression best testing FSALPS individual for Ionosphere dataset with classification accuracy 100% and contains 11 unique features out of 35

(- ion2 (ln (% (% (% (% (* (% ion29 (ln (% (min (ln ion19) (* (% (% ion18 (log10 ion33)) ion19) ion19)) (min (- ion32 ion2) (% (* (% (% ion4 ion3) ion3) ion19) (% (% ion18 (- ion32 ion2)) ion19)))))) ion19) (ln (% (% (% (* (% (% (% (* ion1 ion19) (log10 ion33)) ion6) (max ion21 ion29)) ion19) (ln ion19)) ion29) (max (% (% (* ion1 ion19) ion16) (% (min (ln ion19) (* (% (ln ion19) (* ion1 ion19)) ion19)) (min (- ion32 ion2) (% (% (log10 ion33) (* ion1 ion19)) (log10 ion33))))) ion4)))) ion24) (ln (% (% (% (% (% (min (min (- ion29 ion2) (% (ln ion19) (* ion1 ion19))) (* ion19 ion23)) ion9) ion19) (ln (% (min (- (- (% ion19 (% (ln ion19) (* ion1 ion19))) ion2) ion2) (* (% (- ion32 ion2) ion19) ion19)) (min (min (% (% (% (- ion32 ion2) ion3) ion3) ion3) ion9) ion7)))) ion18) (ln (% (min (- (- ion19 ion2) ion2) (% (% (* (% ion19 (% (* ion1 ion19) (log10 ion33))) ion19) ion32) ion29)) (min (- ion1 ion2) (% (% (* ion1 ion19) (log10 ion33)) (min ion2 (log10 ion33)))))))) (% ion20 ion10))))

Code 5: LISP S-Expression best testing ALPS individual for Ionosphere dataset with classification accuracy 100% and contains 19 unique features out of 35

(* (* (+ enfb53 enfb47) (ln enfb1)) (% (ln enfb53) (max (% (* (max (max (% (* (+ (min enfb53 (* enfb41 enfb51)) enfb47) (ln enfb1)) (+ enfb38 (+ enfb53 (min enfb33 enfb36)))) (min (+ (max (* enfb36 (max enfb58 (% enfb19 enfb11))) (ln enfb18)) (* (* (+ enfb53 enfb47) (ln enfb12)) (max (+ enfb53 enfb47) (max (* enfb41 enfb51) (+ (max (+ (% enfb12 enfb18) (+ enfb38 enfb11)) enfb47) (max (% enfb34 (+ enfb53 enfb57)) enfb22)))))) (max (max (- enfb58 (* (+ (min enfb19 enfb17) (max enfb58

enfb16)) enfb53)) (max (- enfb11 enfb57) (* enfb41 enfb51))) (* enfb41 enfb51)))) (min (min enfb47 enfb11) (+ (- enfb56 enfb11) (max (max (- enfb58 (* enfb36 (% (* enfb1 enfb34) enfb53))) (max (max (- enfb1 enfb41) (+ (- enfb56 enfb41) (max enfb58 enfb16))) (+ (- enfb56 enfb34) enfb1))) (* enfb41 enfb51))))) (- enfb56 enfb11)) (+ (max (% (* (+ enfb38 (+ enfb53 (max (- enfb23 enfb46) (ln (- (- enfb56 enfb11) enfb41))))) (ln enfb58)) enfb35) (min (+ (* (+ enfb53 enfb47) (ln enfb1)) (* (* (+ enfb53 enfb47) (ln enfb1)) (% (ln enfb58) (max (max (% (* enfb23 (- enfb56 enfb11)) enfb1) (min (+ (ln enfb47) (+ enfb10 enfb17)) (+ (+ (ln enfb47) (+ enfb10 enfb17)) (% enfb34 enfb47)))) (min (% (* enfb1 enfb34) enfb53) (+ (ln enfb47) (+ enfb38 enfb11)))))))) (% (- (max (max (min enfb19 enfb17) (min enfb33 enfb36)) (max (- (% (* enfb23 (- enfb56 enfb11)) enfb1) enfb41) (min enfb33 enfb36))) (+ (min (* (+ (% enfb12 enfb18) (+ enfb38 enfb11)) (min (max (+ enfb10 enfb17) (ln enfb53)) enfb17)) (max (max (min enfb19 enfb17) enfb53) (max (max (* enfb34 (- enfb16 enfb34)) (min enfb33 enfb36)) enfb3))) enfb53)) (min enfb56 (* enfb41 enfb51))))) (max (+ (ln (* enfb1 enfb11)) (max (ln (% (* enfb1 enfb34) enfb53)) (+ (% enfb12 enfb18) (max enfb58 enfb16)))) (max (max (% enfb34 enfb47) enfb22) enfb22)))) (min (min (* enfb4 (+ enfb53 enfb47)) (min (* (+ (* (+ (% enfb12 enfb18) (- enfb1 enfb48)) (% (+ (min (* enfb4 enfb22) (max (ln enfb33) (max (max (min enfb19 enfb17) enfb22) enfb22))) (+ enfb53 enfb57)) (max (min (ln enfb47) enfb53) (max (max (* enfb12 enfb52) (- enfb1 enfb48)) (+ (- enfb56 (% enfb34 enfb47)) (+ enfb38 enfb19)))))) (- enfb56 enfb41)) (% (+ (min (* enfb4 (ln (max (% enfb34 enfb47) (% enfb12 enfb18)))) enfb16) (+ enfb53 enfb57)) (* enfb41 enfb51))) (max (min enfb12 (* (+ enfb53 enfb57) (% (max (min enfb19 enfb17) (min enfb33 enfb36)) (max enfb34 (max (ln (% enfb34 enfb47)) enfb36))))) enfb58))) (+ (- enfb56 enfb11) (max enfb58 enfb16))))))

Code 6: LISP S-Expression best testing FSALPS individual for Sonar dataset with classification accuracy 100% and contains 27 unique features out of 61

(+ (- (% (% (- enfb44 (* (min (max (- (% (% (- enfb48 enfb58) (% enfb45 enfb39)) (max enfb50 (% (% (- enfb44 (* (min enfb27 enfb4) enfb37)) (% enfb23 enfb46)) (% enfb45 enfb39)))) (+ (- (% enfb3 enfb45) (% (ln enfb28) (* (max enfb53 enfb38) enfb50))) (- (min enfb24 enfb12) (- (% (+ enfb51 enfb4) (min (+ enfb7 enfb48) (max enfb60 enfb35))) (* (% enfb45 enfb39) (% enfb52 enfb59)))))) enfb45) enfb4) (max enfb53 (- enfb44 enfb58)))) (% enfb45 enfb39)) (* (min enfb27 enfb4) (max

enfb53 (min (* enfb21 enfb44) (max enfb53 (min (% (- enfb44 (* (max (max enfb50 enfb60) enfb60) (max (* (* (ln enfb27) enfb45) enfb26) enfb9))) (* enfb48 enfb9)) (- (% (% (- enfb44 (* (max (* (* enfb2 enfb11) enfb30) enfb60) (max enfb53 (min enfb43 (min enfb12 enfb4))))) (% enfb45 enfb39)) (max enfb50 enfb60)) (% enfb22 (- enfb44 enfb58))))))))) (+ (min (% (% enfb52 enfb59) (* (- (% (% (- enfb44 (* (min enfb27 enfb4) enfb37)) (% enfb45 enfb39)) (max enfb50 enfb60)) (% enfb22 (- enfb44 enfb58))) (min (+ enfb51 enfb4) enfb49))) (- (- (% enfb45 enfb39) (+ (- enfb44 (* (min (* (min (% (- enfb30 enfb45) (* (* enfb2 enfb11) enfb30)) enfb45) (% (- enfb30 enfb45) (* (ln enfb27) enfb30))) (* (% (- enfb27 enfb20) enfb44) enfb9)) (+ enfb20 enfb8))) (% enfb27 (* enfb42 enfb11)))) enfb49)) (* (* enfb48 enfb9) (% (+ (- (% (% enfb52 enfb59) (* (min enfb27 enfb4) (max enfb53 (min (* (% enfb35 (* (max enfb50 (min enfb1 enfb28)) enfb50)) (ln (ln enfb44))) enfb45)))) (min enfb54 (ln enfb7))) (+ (- (min (max enfb53 enfb43) (% enfb22 (* (% enfb56 enfb28) (% (- (% (% (- enfb44 enfb58) (% enfb45 enfb39)) (max enfb50 enfb60)) (% enfb22 enfb12)) enfb28)))) (+ (% enfb52 enfb59) (min (ln enfb44) (- (% (- (% (% (- enfb44 enfb58) (% enfb45 enfb39)) (max enfb50 (min enfb1 enfb28))) (% enfb22 enfb12)) (max (* (* enfb43 enfb7) enfb26) (+ (ln enfb56) (ln (* enfb43 enfb7))))) (+ enfb51 enfb4))))) (* (% (* (+ enfb45 enfb18) (ln (% enfb45 enfb39))) (* (* enfb48 enfb9) (ln enfb7))) (% (% enfb22 (% enfb45 enfb39)) enfb12)))) enfb35)))) (+ (- (% (% (+ enfb45 enfb18) (% enfb45 enfb39)) (min enfb5 enfb4)) (min (% (+ (min (% (% enfb30 enfb59) (+ (% (max (% (- enfb27 enfb20) enfb44) enfb32) (% (- enfb44 enfb58) (* (max enfb16 (min enfb24 enfb12)) enfb10))) enfb15)) enfb45) enfb18) (* (ln enfb7) (* enfb46 enfb22))) (% (+ (- (% (% enfb52 enfb59) (* (min enfb27 enfb4) (max enfb53 (min (* enfb21 enfb44) enfb45)))) (max (max (% (% enfb45 (% enfb45 (% enfb45 enfb39))) (min enfb5 (+ enfb51 enfb4))) enfb31) enfb31)) (+ (- (min enfb54 (% enfb22 (* (- enfb48 enfb58) (% (max (% (- enfb27 enfb20) enfb44) (* (- enfb15 enfb12) (max enfb53 enfb43))) (% (- enfb44 (- (% enfb56 enfb28) enfb48)) (% enfb22 enfb12)))))) (+ (% enfb35 (* (max enfb53 enfb38) enfb50)) (min enfb44 (* (% (* (+ enfb45 enfb18) (ln (% enfb45 enfb39))) (* (* enfb48 enfb9) (ln enfb7))) (% (% enfb22 (% enfb45 enfb39)) enfb12))))) (* (% (* (+ enfb45 enfb18) (ln (% enfb45 enfb39))) (* (* enfb48 enfb9) (ln enfb5))) (% (% (+ enfb45 enfb18) (% enfb45 enfb39)) enfb12)))) enfb35))) (* (% (* (+ (min (% (% enfb30 (% (- enfb44 (- (% enfb56 enfb28) enfb48)) (% enfb22 enfb12))) (+ (% (max (% (- enfb27 enfb20) enfb44) (* (- enfb15 enfb12) (max enfb53 enfb43))) (% (- enfb44 enfb58) (% enfb45 enfb39))) enfb15)) enfb45) enfb18) (ln enfb28)) (* (max enfb50 enfb60) (ln enfb50))) (% (% enfb52 enfb59) enfb12))))

Code 7: LISP S-Expression best testing ALPS individual for Sonar dataset with classification accuracy 100% and contains 45 unique features out of 61

Appendix B

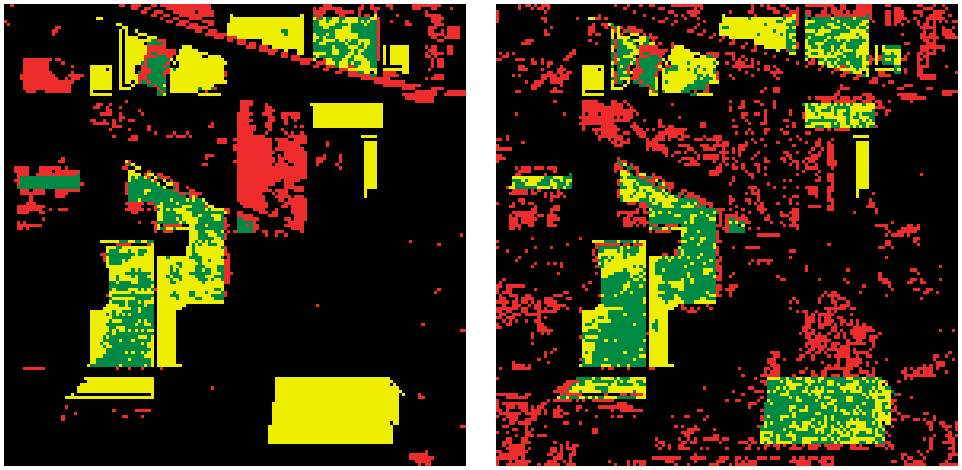
Hyperspectral

B.1 Performance Analysis

In this section, we will do further analysis of performance plots for the Indian Pines hyperspectral dataset.

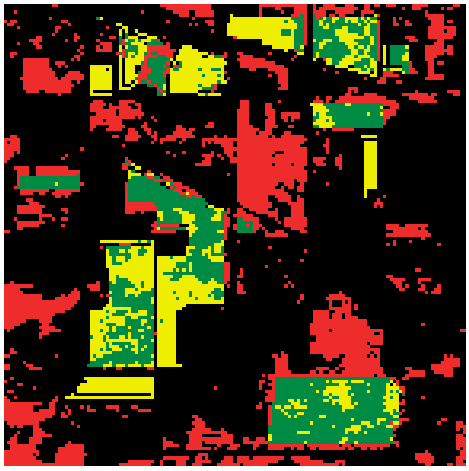
B.1.1 Classification Accuracy

All the algorithms did not record high classification accuracy for the soybean–mintil class. The binary classification task for soybean–mintil involves 18.30% of TP and 81.70% of TN. GP could detect all negative cases without detecting crop features and still score high classification accuracy. We see this happening with canonical GP that scores 90.45% of TNs but only 26.69% of TPs. ALPS and FSALPS produced much more generalized solutions with above average scores for both TP and TN. The poor performance is likely due to the uneven number of positive and negative examples used for training.



(a) Canonical

(b) ALPS

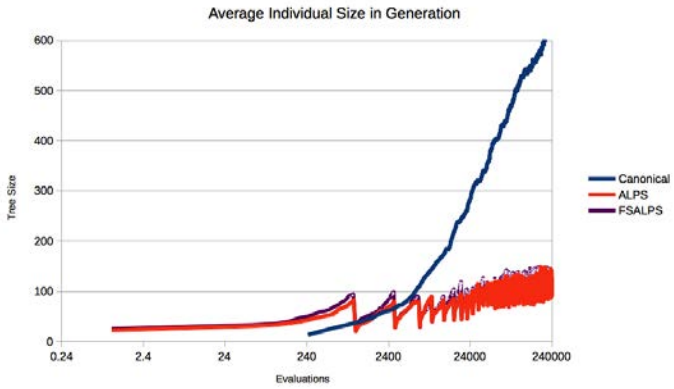


(c) FSALPS

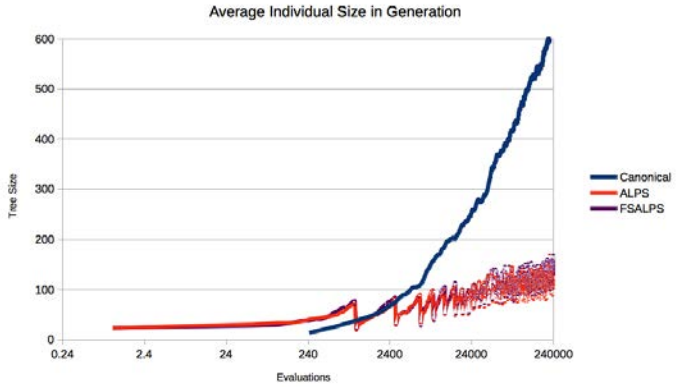
Figure B.1: Performance plot for Soybean–mintil with (a) TP detection : 26.69% , TN detection : 90.45% overall performance : 78.78% (b) TP detection : 52.47% , TN detection : 82.94% overall performance : 77.36% (c) TP detection : 52.70% , TN detection : 79.14% overall performance : 74.30%

B.1.2 Memory Usage and Bloat Control

A logarithmic plot of tree size reveals a sharp rise in memory usage by canonical GP. Canonical GP is often susceptible to bloat and large tree sizes. Although no explicit bloat control mechanism was implemented, ALPS and FSALPS succeeded in evolving simple but effective trees. Tree size is directly proportional to evaluation time hence ALPS and FSALPS are more resource efficient.



(a) Corn–notill



(b) Soybean–mintil

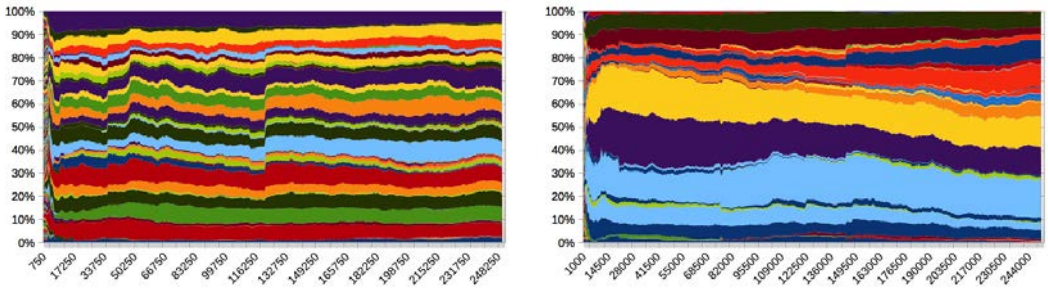
Figure B.2: Space analysis using logarithmic plot for generational growth in tree size

B.2 Feature Analysis

In the plots shown in Figure B.3 –B.8, a color band represents how a unique feature evolves through the population. Canonical GP (see Figure B.3 and B.6) heavily relies on the features selected at initialization for the construction of new individuals. However, new features might be introduced when performing a mutation operation. Mutation involves random construction of a sub–tree that is rooted on a randomly selected node on a GP individual. In order to avoid "random walk"a very low mutation probability was used, which reduces the chance of new features introduced into the population after initialization. Although canonical GP offers a stable feature selection, the tendency of restricting search to a localized area in the search space is very high which could leave a great part of the search space unexplored. We also discovered that feature subset of the best training individual is usually a subset of the features selected during initialization.

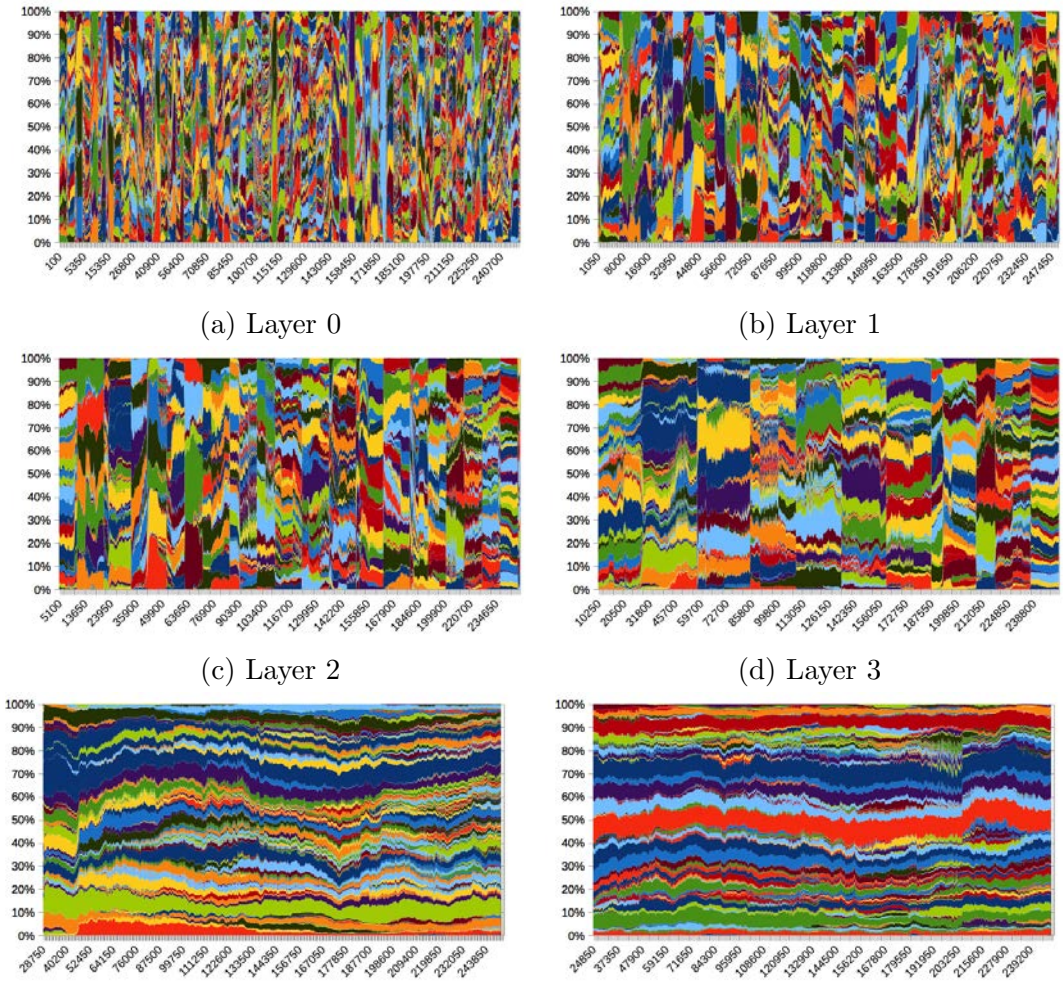
The ALPS algorithm attempts to overcome this inherent problem in canonical GP by regularly introducing new individuals into the bottom layer. An individual is always initialized by randomly selecting terminal symbols (features) from the original feature subset. This unique strategy means there is a high probability to introduce new feature into the ALPS population. Inter layer breeding allows these new features to move up the layers. Evolution pressure favors individuals with relevant features — leading to feature selection (see how a very random Layer 0 gradually filters to relevant features in higher layers of Figure B.4 and B.7). On the other hand, ALPS ability to explore a larger part of the fitness landscape reduces its ability to perform feature selection. Higher layers are more stable than lower layers since they are less disruptive. The regular introduction of new features means the average individual usually has more features than canonical GP, which led to the introduction of the FSALPS strategy.

FSALPS (see Figure B.5 and B.8) performs a directed feature selection by using evolved probability values for each feature to determine selection of features during initialization of each FSALPS bottom layer. These probability values are translated from feature frequency counts obtained by enumerating features in the population. We do not completely eliminate each feature, but rather, evolution favors highly relevant features while reducing (but not eliminating) less relevant features. $Layer0$ of Figure B.5 and B.8 gradually transitions from equal probability of selection for all features to evolved probability values. No feature is completely eliminated to avoid chances of restricting search in the solution space and landing on a mediocre solution. The probability values are also used during the construction of sub–trees for mutation operations. The directed feature selection process in FSALPS is visible in "$Layer0$"of Figure B.5 and B.8. In FSALPS, higher layers are more stable than lower layers since they are less disruptive. This stabilization is highly expressed in FSALPS than the ALPS and canonical GP.



(a) Overall best (b) Minimum features

Figure B.3: Corn–notil: Percentage contribution of each feature in canonical run with (a) overall best individual had 39 features and 92.51% classification accuracy (b) minimum features had best individual with 18 features and 91.96% classification accuracy.



(a) Layer 0

(b) Layer 1

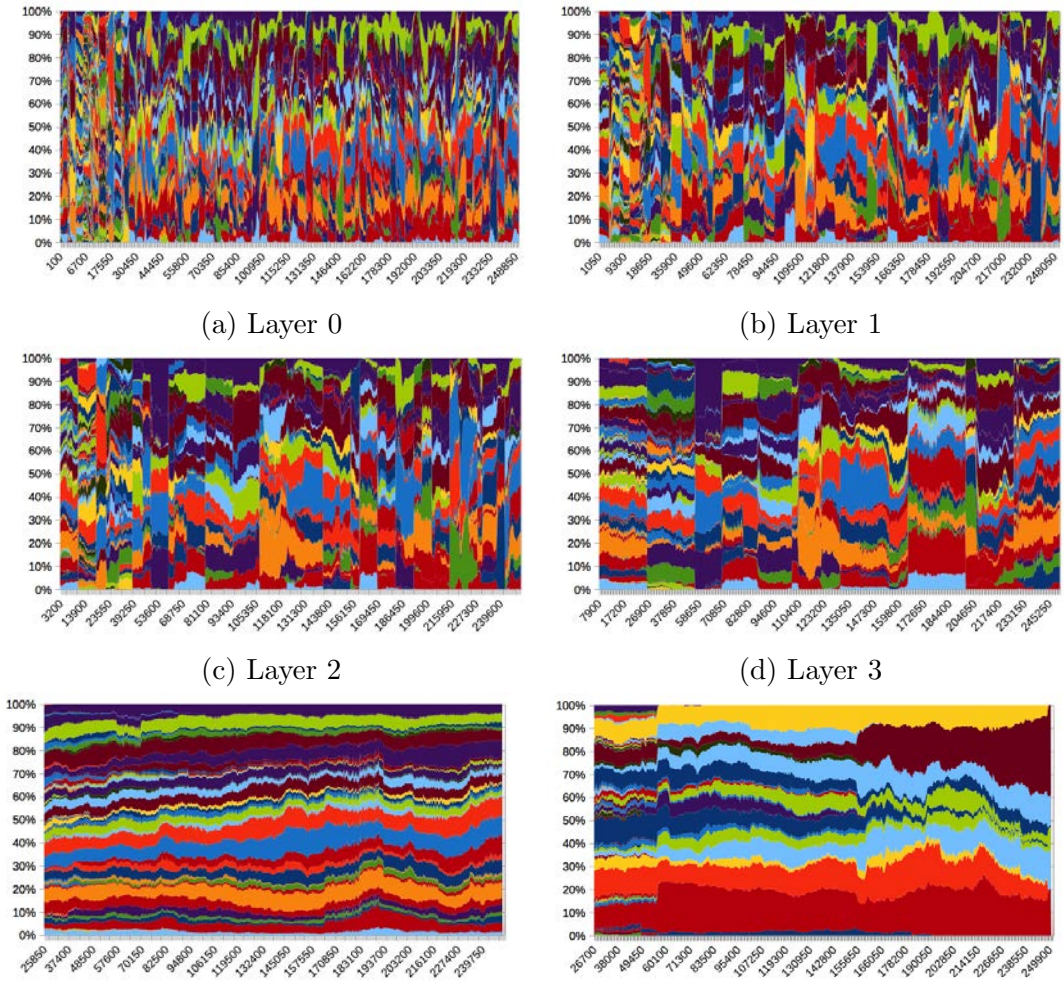
(c) Layer 2

(d) Layer 3

(e) Layer 4

(f) Layer 4

Figure B.4: Corn–notil: Percentage contribution of each feature per layer in the ALPS–run with (a–e) Layer 4 of best individual having 48 features and 93.69% classification accuracy (f) Layer 4 of individual with minimum features had 31 features and 87.78% classification accuracy.



(a) Layer 0

(b) Layer 1

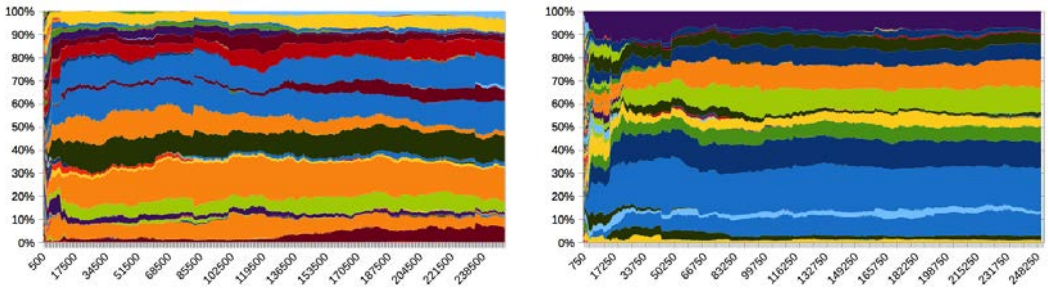
(c) Layer 2

(d) Layer 3

(e) Layer 4

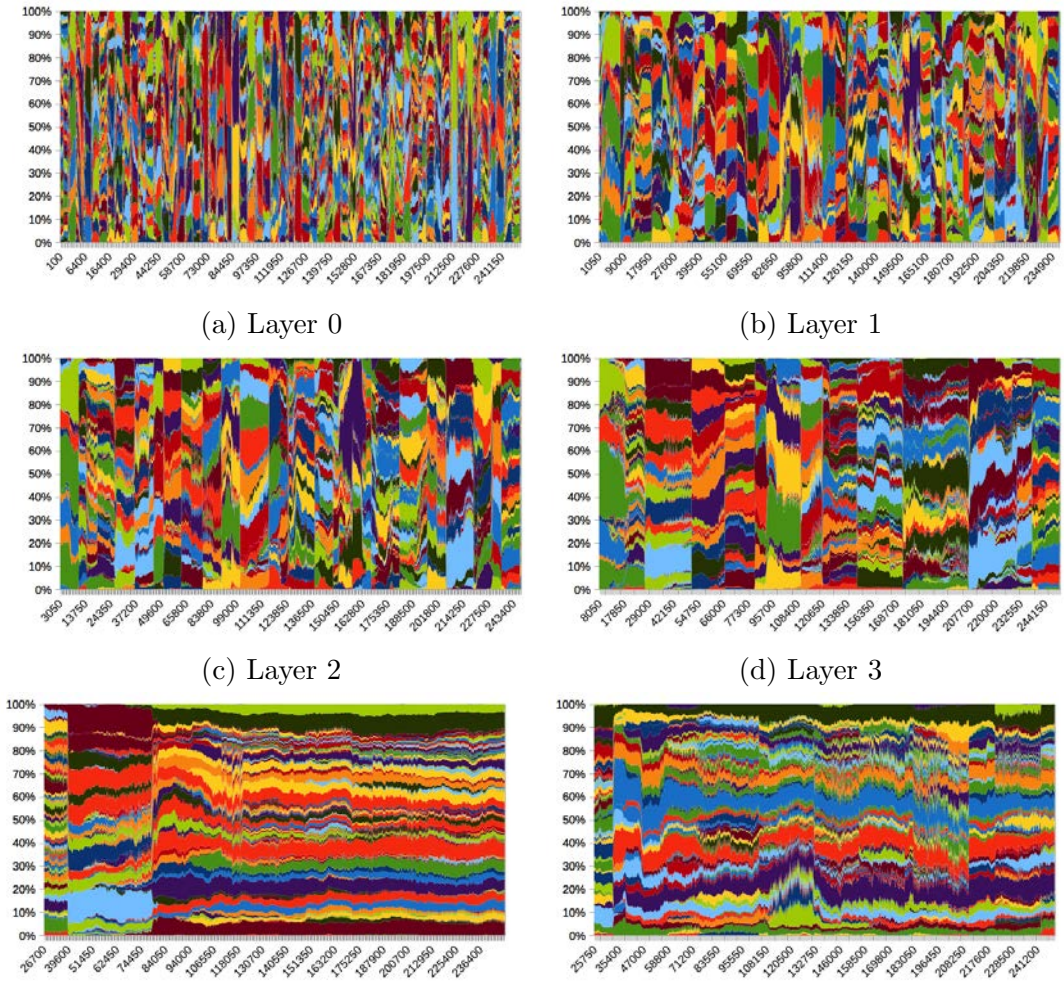
(f) Layer 4

Figure B.5: Corn–notil: Percentage contribution of each feature per layer in the FSALPS–run with (a–e) Layer 4 of best individual having 35 features and 92.16% classification accuracy. (f) Layer 4 of individual with minimum features had 7 features and 80.48% classification accuracy.



(a) Overall best (b) Minimum features

Figure B.6: Corn–notil: Percentage contribution of each feature in canonical run with (a) overall best individual had 21 features and 78.78% classification accuracy (b) minimum features had best individual with 17 features and 64.91% classification accuracy.



(a) Layer 0

(b) Layer 1

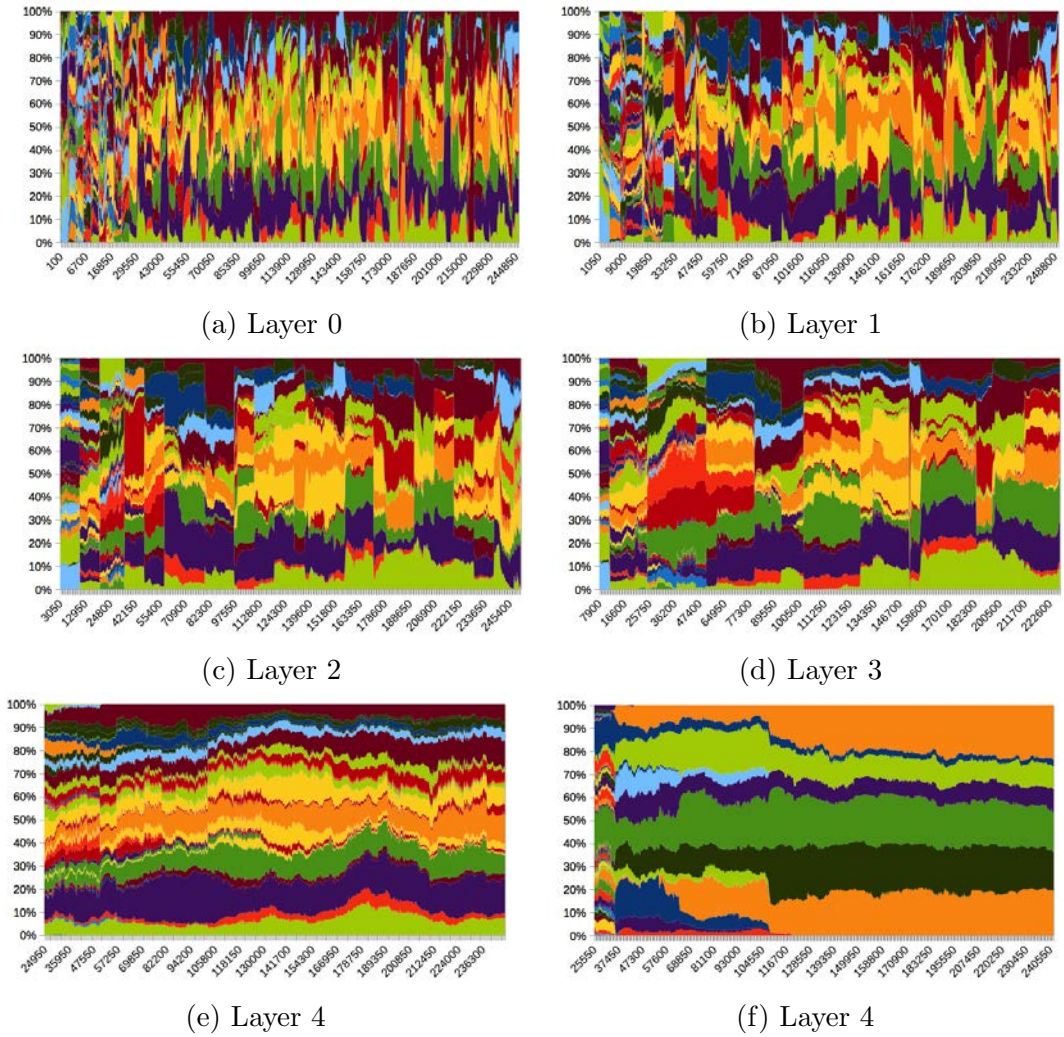
(c) Layer 2

(d) Layer 3

(e) Layer 4

(f) Layer 4

Figure B.7: Soybean–mintil: Percentage contribution of each feature per layer in the ALPS–run with (a–e) Layer 4 of best individual having 56 features and 77.36% classification accuracy (f) Layer 4 of individual with minimum features had 29 features and 61.08% classification accuracy.



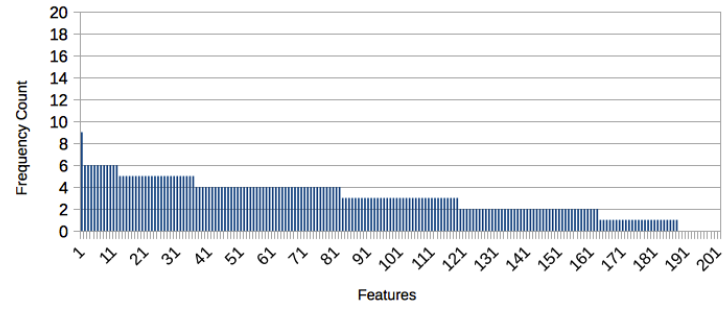
(a) Layer 0 (b) Layer 1

(c) Layer 2 (d) Layer 3

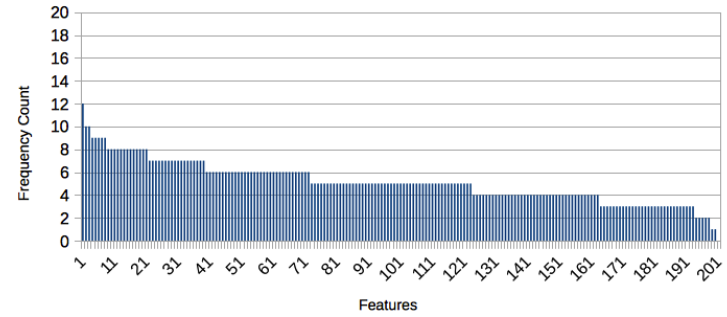
(e) Layer 4 (f) Layer 4

Figure B.8: Soybean–mintil: Percentage contribution of each feature per layer in the FSALPS–run with (a–e) Layer 4 of best individual having 20 features and 74.30% classification accuracy (f) Layer 4 of individual with minimum features had 7 features and 67.78% classification accuracy.

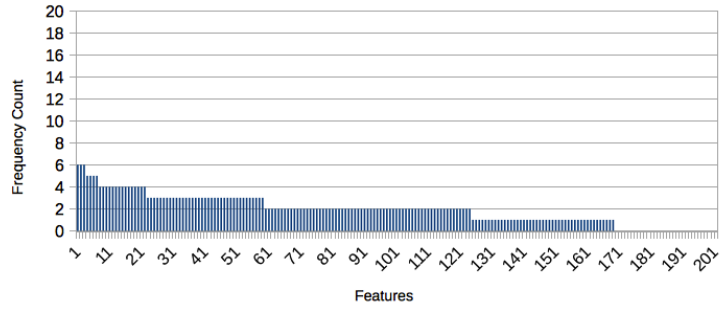
Figure B.9 shows a plot of feature frequency in all 20 runs. The feature counts are made from the best training individual used in testing. We used this to test consistency in features selected by any of the algorithms for all 20 runs. FSALPS has a higher consistency followed by canonical GP and then ALPS. The plots show higher number of features in ALPS. It also reveals a very random process in ALPS and canonical GP for each run especially canonical GP. Canonical GP was found to have fewer features than ALPS however each separate run begins from a new fitness basin with new features. Thus we see completely new features (in Figure canonical) introduced for each run. This also proves a local search happening in canonical GP.



(a) Canonical



(b) ALPS



(c) FSALPS

Figure B.9: Number of times features were used in all 20 runs for Corn–notill in the Indian Pines hyperspectral dataset

B.3 Solution Trees

(min (max (+ b61 -0.39976105) (+ (+ (* b195 b141) (- b118 b90)) (+ (* (% b40 b117) b68) (- (* (% (- (% (+ (- (% (% b40 b117) b40) (+ (* b91 (* b195 b141)) (+ (% b46 b159) (max b199 b152)))) (+ (max b199 b152) (* (% (% b23 b47) (% b23 b47)) (- b118 b90)))) (% b23 b47)) -0.39976105) (% (+ (+ (- b29 b70) (* (min b124 (% (% b91 b43) (% b23 b47))) (% b91 b43))) (+ (- b161 b135) (* (min b124

(- b118 b90)) (+ b81 -0.39976105)))) (% b91 b43))) b40) (+ (- (% (% b40 b117) (* b195 b141)) (* (% (* (% (% (% b46 b159) (* b195 b141)) b104) (% b46 (* (% b40 b47) b68))) (* b162 b194)) (+ (- b118 b90) -0.39976105))) -0.39976105))))) (min (+ (- (- (* b195 b141) (% (min (* b124 b141) (% (* (% (* (% (% (% b40 b117) (* b195 b141)) b104) (% b90 b110)) (* b195 b141)) (+ (- b118 b90) -0.39976105)) (+ (min b1 b151) b141))) (% b23 b47))) (- b29 b70)) (min (min (% (+ (- (- (* b195 b141) (+ (% (* b195 b141) (% b23 b47)) (+ (% b46 b159) (+ (% b46 b159) (max b199 b152))))) (* (% b23 (- (% (* b195 b141) b40) (+ (max (min b1 b151) (min b1 b151)) (% b91 b43)))) b68)) (% (% (% b46 b159) (- b118 b90)) b104)) (* b195 b141)) (min (% (max b47 (* b115 b194)) (* b195 b141)) (* (% (% b91 b43) (% (+ (- (% (% b40 b117) (* b195 b141)) (% b40 b47)) (+ (* b195 b141) b141)) (% b23 b47))) (% b46 b159)))) (* (% (% b91 b43) (% (* (% (% b91 b43) (% (+ (- (% (% b40 b117) (* b195 b141)) (+ (max (+ b74 -0.39976105) (* b162 b194)) (+ (min b1 b151) b141))) (+ (- b161 b135) (* (min b124 (* (% b40 b117) b68)) b195))) (% b23 b47))) (% b46 b159)) b47)) (+ (- (- b118 b90) (+ (- (- b118 b90) (- (* b195 b141) b44)) -0.39976105)) -0.39976105)))) (min (min (* (% (- (min b124 (% (* (% (- (% (% b40 b117) (* b195 b141)) (+ (max (+ b74 -0.39976105) (* b162 b194)) (+ (% (% b91 b43) (* b195 b141)) b141))) (* b162 b194)) (+ (% (* b195 b141) (+ (- b118 b90) b154)) (min b124 b46))) b74)) b90) b10) (min (% (% b40 b117) (* (% -0.39976105 (- b118 b90)) (+ -0.39976105 (min b124 (* b195 b141))))) b47)) (min (+ (- (% (* b195 (* b115 b194)) b92) -0.39976105) (* (+ (min b1 b151) b141) b185)) (+ (- (* b195 b141) b44) (min (* (* (* (* (% (% (% b40 (% b91 b43)) (* b195 b141)) b104) (% b90 b110)) b194) (% b91 (+ (- (% (% b40 b117) b40) (min b1 b151)) (+ (- (* (% (% b46 b159) (- b118 b90)) b141) (% (% b91 b43) (- b161 b135))) (% b90 b110))))) b116) (min (% (% b40 b117) (* b195 b141)) b141))))) (- (% (+ (max b199 b152) (+ (+ (* b195 b141) (* (+ (min b1 b151) b141) (% (+ (- (% (* b195 b141) (* b195 b141)) (+ (max (+ b74 -0.39976105) (* b162 b194)) (- b118 b90))) (% (+ b43 (% (- b118 b90) b43)) (% b46 (* (% b40 b47) b68)))) (min (- (* b195 b141) (* (% b40 b47) b68)) b47)))) (- (% (min (min (* b124 b141) (% b23 b149)) b47) b47) (+ (max (+ b74 -0.39976105) (* (min b124 (% (% b91 (* b162 b194)) (% b23 b47))) (% b91 b43))) (+ (min b1 b151) b141)))))) (min (* (* (- (% (+ (- (% (% b40 b117) b40) (+ (* (% (% b91 b43) (% b23 b90)) (* b195 b141)) (+ (% b46 b159) (max b199 b152)))) b104) (% b23 b47)) -0.39976105) (- (% (% (min b1 b151) b43) (% b23 b149)) b90)) (+ (- b118 b90) -0.39976105)) b49)) (max (+ (+ (* (* b124 b141) (% b194 (% (% (+ (min b1 b151) b141) (- b118 b90)) b104))) (* (- b118 b90) (% (% b40 b117) (% b91 b43)))) (+ (* (* b195 b141) (- b118 b90)) (* (% (% (% b46 b159) (* b195 b141)) b104) (min (% b46 b159) b47)))) (* (* (% (% b91

b151) (% b90 (% b23 (- (* b195 b141) (% b91 b43))))) (% b46 b159)) b194))))))

Code 1:LISP S-Expression best testing canonical individual for Corn–notill dataset with classification accuracy 92.52% and 39 unique features out of 201 features

(min (% (- b2 b91) (min (% (- b164 (+ (% b97 b157) (* (+ b3 (- b10 (% b145 b20))) (min (* b61 b141) (min (* b78 (- (- b99 b93) (min (* b61 b141) b164))) (% b97 b157)))))) (+ (- (+ (% b145 b20) (- (- b30 b46) (+ (+ b3 (- (* b61 b141) (% b145 b20))) (+ b65 b22)))) (- (+ (% (min (min (+ b96 (- (* b61 b141) (* (+ (- b55 b199) (+ b105 b9)) (* (max b75 b98) (- b98 b55))))) (% (+ b187 (- b2 b91)) (* b61 b141))) b21) b20) (- (- b2 b91) (+ (% b145 b10) (% (+ b147 b82) (+ b166 b192))))) (- b164 (+ (* b61 b141) b141)))) (+ b169 (min b88 (+ b42 (- (- (* b61 b141) (% b145 b20)) (% b97 b157))))))) (* (- b30 b46) (* b78 (+ (% b150 b198) (- b130 b141)))))) (min (min (% (+ (+ (% (+ (* (+ b187 b169) (+ (- b124 b27) (* (+ b3 (% b145 b20)) (% b106 (% b106 b13))))) b124) (min (* (- b124 b27) b66) (- (+ b187 b169) (% b145 b20)))) b10) (% (min (min (* b61 b141) (min (min b84 (min (- b2 b91) (- (- (- b2 b91) (- b27 b52)) b96))) (% b150 b198))) (+ b65 b22)) b157)) (min (min (- (- b30 b46) (- (- b99 b93) (% b82 b154))) (min (+ (+ b76 b177) (- (* b61 b141) (% b145 b20))) (% (+ b187 (- b2 b91)) (+ (min (- b23 b46) (- b2 (% b68 b198))) (% (* (* (% b106 (% b106 b13)) b66) (% b145 b20)) (+ b166 (min (+ b147 b82) b46))))))) b21)) (min (% (min (min (* b61 b141) (min (min b84 (min (- b2 b91) (- (- (- b2 b91) (- b27 b52)) b96))) (- (- b164 b27) b96))) (+ b187 b169)) b157) (+ (- (+ (% b145 b20) (- (- b30 b46) (+ (% b145 b20) (- (- b124 b27) b96)))) (- b164 (+ (* b61 b141) (* (+ b3 (- (* b61 b141) (% b145 b20))) (min (min (- b2 b91) (- (- b124 b27) b96)) (+ b187 b169)))))) (max b124 b116)))) b61))

Code 2: LISP S-Expression best testing ALPS individual for Corn–notill dataset with classification accuracy 93.69% and 48 unique features out of 201 features

(- (min (min (max (- (- (+ b66 b16) (% b50 b149)) (+ (- (% b21 b169) (% b11 b59)) (- b72 b160))) (- (* (min (max b34 (+ (* (% (+ (* (+ b11 (+ b149 b34)) (- (- b141 b49) (max b189 b93))) b90) (- b68 b89)) (- (max b68 b189) (* b11 b60))) b90)) (min b11 (min b40 (- (max b34 b189) (* b59 b60))))) (- (* (min (max (- b141 b49) (+ (* (+ b11 (+ b149 b34)) (* b34 (- (- b141 b49) (max b189 b93)))) b90)) (+ b11

(+ b149 b34))) (min (min (- (max b34 b189) (* b59 b60)) (- b176 b60)) b149)) b11)) b11)) (min b6 b13)) (max (% (+ (- (min (min b21 (max b13 (* (- (min b23 b6) (% b90 b89)) (* (- b176 b60) b60)))) (max (% (+ (* (+ b11 (+ b149 b34)) (- (- b141 b49) (max b189 b93))) b90) (- b68 b89)) (min (+ (* (- b176 b60) (- (- b141 b49) (max b189 b93))) b90) (min (min b40 b62) (- (+ b68 b90) (% (min (max b107 (min b23 b62)) b72) (max (- b176 b60) (- (- b141 b49) (max b189 b93)))))))))) (+ (* (+ (* (- b176 b60) b104) (min b91 b62)) (min (max (+ (* (* b59 b60) (min (min (- (max b34 b189) (* b59 b60)) (- b176 b60)) b149)) (% (min b141 (min (min b75 b151) b62)) (% (* b59 b60) (- b141 b49)))) (+ (* (+ b11 (+ b149 b34)) (- b141 (max b189 b93))) b90)) b189)) (% b62 (% b62 (max (- b176 b60) (min b40 (min b6 (% b73 b137)))))))) b90) (- b68 b89)) (% b50 (- b68 (+ b59 (% b11 b6))))))) (+ (* b34 (- (- b141 b49) (max b189 b93))) (% (min (min (min b23 b62) (min b21 b23)) (% b23 b34)) (% (min (min (% (min (min b75 b151) b62) (- b141 b49)) b62) (+ (% (min (max (% b35 (- (max (% b21 b169) (* (+ b149 b34) (min (+ b66 b16) (% b73 (- b141 b49))))) b89)) (min (% b49 b141) (- b176 b60))) (+ (% b73 b137) b34)) (max (+ b35 b62) b176)) (% (min b177 b46) (% (- b68 b89) (min (+ (* (+ b11 (+ b149 b34)) (- (- b141 b49) (max b34 b189))) b90) (max (% (- b75 b149) (- (min (max (- b176 b60) (* b59 b60)) b149) (% (min b141 (min (* b59 b60) b62)) (% (* b59 b60) (- b141 b49))))) (- b21 b160))))))) (min (% (min (min b11 (min (- b68 (- b89 b34)) (max (% (+ (* (+ b11 (+ b149 b34)) (- (- b141 b49) (max b189 b93))) b90) b62) (- b68 b89)))) b40) b62) (+ b35 b62))))))

Code 3: LISP S-Expression best testing FSALPS individual for Corn–notill dataset with classification accuracy 92.16% and 35 unique features out of 201 features